

EASTERN SWITZERLAND UNIVERSITY OF APPLIED SCIENCES

BACHELOR THESIS

Completeness Estimation of OpenStreetMap POI Data Using Machine Learning Approaches

Authors:

Marco CRISAFULLI
Dominic MONZÓN

Advisor:

Prof. Stefan F. KELLER

Co-Advisor:

Nicola JORDAN

Institute for Software
Department of Computer Science
Campus Rapperswil-Jona

June 18, 2021



[HTTPS://EPRINTS.OST.CH/940](https://eprints.ost.ch/940)

Copyright © 2021 Marco CRISAFULLI, Dominic MONZÓN

June 18, 2021

EASTERN SWITZERLAND UNIVERSITY OF APPLIED SCIENCES

Abstract

Campus Rapperswil-Jona
Department of Computer Science

Completeness Estimation of OpenStreetMap POI Data Using Machine Learning Approaches

by

Marco CRISAFULLI
Dominic MONZÓN

As OpenStreetMap (OSM) gains traction and is considered a viable alternative to service providers like Google Maps, the question of the quality of the provided data becomes increasingly important. A key factor for the quality of geographical data is the completeness of entities that are included or omitted in a dataset. And currently, there is no general solution to determine it. The vision of this project is to lay the groundwork for an approach with an open-source tool that can be used by the community and by users to check desired areas for completeness.

This work aims to estimate intrinsically - i.e., without comparing to a 'golden dataset' - the number of Points of Interest (POIs) in a defined area. These values compared to the number of existing POIs act as an indicator for completeness. The nature of the problem and the size of available data is predestined for machine learning (ML) methods. An initial model was trained based on high-resolution imagery (orthophotos). It showed that there are relationships that can be detected by ML algorithms. Thus, a model was trained using only intrinsic data provided by OSM. Under the assumption that the training and validation areas are completely mapped, the implemented model performs well enough to show a trend where entities are missing.

The results are visualized in a color-coded grid showing the areas which are predicted to either be complete, improvable, or incomplete. As it is trained on data in Swiss cities it works best for urban areas in Switzerland and neighboring countries because of the geographic and demographic similarities. By use of re-training the model it is possible to predict other areas. One drawback of the intrinsic approach is that a certain amount of existing data is needed to make a prediction. Further, the quality of the prediction itself can only be measured on the assumption that the training and validation areas are well mapped. In conclusion, we provide a model which estimates the completeness of an area and indicates if further investigation is needed.

Keywords: OpenStreetMap, Machine Learning, Completeness

OST - OSTSCHWEIZER FACHHOCHSCHULE

Zusammenfassung

Campus Rapperswil-Jona
Studiengang Informatik

Abschätzung der Vollständigkeit von OpenStreetMap POI-Daten mit Hilfe von Machine Learning Ansätzen

von
Marco CRISAFULLI
Dominic MONZÓN

Die Beliebtheit von OpenStreetMap (OSM) als Alternative gegenüber etablierten Dienstleistern wie Google Maps steigt. Daher wird die Frage nach der Qualität der bereitgestellten Daten immer wichtiger. Ein Schlüsselfaktor für die Qualität von geografischen Daten ist die Vollständigkeit der Entitäten, welche vorhanden sind oder bewusst weggelassen wurden. Derzeit gibt es keine universelle Lösung, um die Vollständigkeit zu bestimmen. Die Vision dieses Projekts ist es, mithilfe eines Open-Source-Projekts die Grundlage für eine Lösung zu schaffen, welche von der Community genutzt und weiterentwickelt werden kann.

Ziel dieser Arbeit ist es, intrinsisch - d.h. ohne Vergleich mit einem "goldenen Datensatz" - die Anzahl der Points of Interest (POIs) in einem definierten Gebiet zu schätzen. Diese Werte, verglichen mit der Anzahl der vorhandenen POIs, dienen als Indikator für die Vollständigkeit. Die Art des Problems und der Umfang der verfügbaren Daten eignen sich bestens für Methoden des maschinellen Lernens (ML). Ein erstes Modell wurde auf Basis von hochauflösendem Bildmaterial (Orthofotos) trainiert. Es zeigte sich, dass es Zusammenhänge gibt, die von ML-Algorithmen erkannt werden können. Daher wurde ein Modell trainiert, das nur die von OSM bereitgestellten und daraus berechneten Daten verwendet. Unter der Annahme, dass die Trainings- und Validierungsgebiete vollständig kartiert sind, erbringt das implementierte Modell eine ausreichende Leistung, um einen Trend zu zeigen, in welchen Gebieten Entitäten fehlen und wo es gegebenenfalls noch sinnvoll wäre, weitere Ressourcen zu investieren.

Die Ergebnisse werden in einem Ampel-ähnlichen farbcodierten Raster visualisiert, welches die Gebiete anzeigt, die entweder als vollständig, verbesserungswürdig oder unvollständig eingeschätzt werden. Da das Modell basierend auf Daten von Schweizer Städten trainiert wurde, funktioniert es am besten für urbane Gebiete in der Schweiz und den Nachbarländern. Durch erneutes Trainieren des Modells ist es möglich, andere Gebiete mit anderen geografischen und demografischen Gegebenheiten vorherzusagen. Ein Nachteil des intrinsischen Ansatzes ist, dass eine gewisse Menge an vorhandenen Daten benötigt wird, um eine Vorhersage zu treffen. Außerdem kann die Qualität der Vorhersage selbst nur unter der Annahme gemessen werden, dass die Trainings- und Validierungsgebiete gut abgebildet sind.

Acknowledgements

First and foremost, we are extremely grateful to our advisors Prof. Stefan F. Keller and Nicola Jordan for their invaluable advice, challenging questions and healthy criticism during our work. This project would not have been possible without their help and their passion for the same. We would also thank our families and partners for the tremendous patience and emotional support during these challenging times. Lastly, we want to thank our fellow students and friends, for the feedback and valuable support.

Contents

Abstract	iii
Acknowledgements	vii
Contents	ix
List of Figures	xi
List of Tables	xiii
Task Description	xv
Management Summary	xvii
1 Technical Report	1
1.1 Introduction	1
1.1.1 Problem Definition, Vision	1
1.1.2 Goals	1
1.1.3 General Conditions, Definitions, Delimitations	2
1.1.4 Method	2
1.2 State of the Technology	3
1.2.1 Existing Solutions and Standards	4
1.2.2 Use Cases	6
1.2.3 Non-Functional Requirements	7
1.3 Evaluation	8
1.3.1 Development Environment and Libraries	8
1.3.2 Machine Learning Framework	9
1.3.3 Demo	10
1.4 Implementation Concept	12
1.4.1 Base Model Using Orthophotos	12
1.4.2 Data Preparation Pipeline	13
1.4.3 Extending Data Preparation	15
1.4.4 Base Model Using Only OpenStreetMap Data	15
1.4.5 Improved Models	15
1.4.6 Experiment Tracker	16
1.4.7 Blurring	17
1.5 Results	18
1.5.1 Base Model	18
1.5.2 Experiments to Improve Results	19
1.5.3 Improved Model Quality	24

1.5.4	Final Classification	28
1.5.5	Limitations	30
1.5.6	Target Achievement	30
1.5.7	Demo	31
1.6	Outlook, Further Development	31
1.6.1	Data Enhancements	31
1.6.2	Performance Optimizations	32
	Bibliography	33
	Glossary	35
	Acronyms	37
A	Completeness of Shops for Selected Areas	A-1
B	Comparison of Actual and Predicted Shop Counts for Selected Areas	B-1
C	Software Documentation	C-1
D	Project Management	D-1
E	Time Tracking	E-1

List of Figures

1	Schema of the Data Preparation Pipeline	xviii
2	Estimated Completeness of Shops for Rapperswil	xviii
1.1	Mockup of Visual Presentation of Results With 3 Color Categories	2
1.2	Visualization of KNN and Density Calculation	3
1.3	Overlap Between Official and OSM Datasets	5
1.4	Completeness of Gastronomy Objects in Bern Estimated by OSMCross	6
1.5	Jupyter notebook screenshot	8
1.6	Screenshot of Demo Optimized for FullHD (1920x1080px) Screens	11
1.7	Prediction of Number of POIs for Rapperswil Based on Aerial Images	13
1.8	Schema of the Data Preparation Pipeline	14
1.9	Left: Blurred Results, Right: Unblurred Results	17
1.10	Difference Between Predicted and Actual Shop Count for Base Model	19
1.11	Training and Validation Loss by Experiment	20
1.12	Metrics by Optimizer	21
1.13	Mean Absolute Error by Optimizer (Adam and Lamb in Focus)	22
1.14	Mean Absolute Error by Batch Size for LAMB	23
1.15	Mean Absolute Error by Batch Size for Adam	23
1.16	Prediction for Rapperswil	25
1.17	Prediction of Shops for Bern	25
1.18	Comparison Actual and Predicted Shop Count for Bern	26
1.19	Prediction for Luzern	26
1.20	Prediction for Luzern	27
1.21	Prediction for Rapperswil (Zoomed, 100m Grid)	27
1.22	Prediction for Rapperswil (Zoomed, 200m Grid)	28
1.23	Estimated Completeness of Shops for Rapperswil	29
A.1	Estimated Completeness of Shops for Rapperswil	A-1
A.2	Estimated Completeness of Shops for Bellinzona	A-2
A.3	Estimated Completeness of Shops for Bern	A-2
A.4	Estimated Completeness of Shops for Brugg	A-3
A.5	Estimated Completeness of Shops for Burgdorf	A-3
A.6	Estimated Completeness of Shops for Geneva	A-4
A.7	Estimated Completeness of Shops for Lausanne	A-4
A.8	Estimated Completeness of Shops for Lucerne	A-5
A.9	Estimated Completeness of Shops for Solothurn	A-5
A.10	Estimated Completeness of Shops for Zürich	A-6
B.1	Comparison Actual and Predicted Shop Count for Rapperswil	B-1

B.2	Comparison Actual and Predicted Shop Count for Bellinzona	B-2
B.3	Comparison Actual and Predicted Shop Count for Bern	B-2
B.4	Comparison Actual and Predicted Shop Count for Brugg	B-3
B.5	Comparison Actual and Predicted Shop Count for Burgdorf	B-3
B.6	Comparison Actual and Predicted Shop Count for Geneva	B-4
B.7	Comparison Actual and Predicted Shop Count for Lausanne	B-4
B.8	Comparison Actual and Predicted Shop Count for Lucerne	B-5
B.9	Comparison Actual and Predicted Shop Count for Solothurn	B-5
B.10	Comparison Actual and Predicted Shop Count for Zürich	B-6
C.1	nbdev generated code documentation	C-2
D.1	Organizational Structure	D-2
D.2	Project Lifecycle	D-3
E.1	Work by Label	E-2
E.2	Cumulated Work Over Time	E-2
E.3	Seperate Work Over Time	E-3

List of Tables

1.1	Evaluation TensorFlow with Keras	9
1.2	Evaluation PyTorch with fast.ai	9
1.3	Existing Experiment Trackers	16
1.4	Tolerance for Prediction	28
1.5	Classification of Shops per Area	30
D.1	Links and Tools	D-1
D.2	Milestones and Deadlines	D-1
D.3	Risk Analysis	D-4
D.4	Risks Incurred	D-4
E.1	Five Most Over- and Underestimated Work Items	E-1

Task Description



Completeness Estimation of OpenStreetMap POI Data Using Machine Learning Approaches

Bachelorarbeit im Frühlingsemester 2021, Studiengang Informatik

Einleitung

Points-of-Interest (POI) haben einen hohen Marktwert und OpenStreetMap ist zur ebenbürtigen freien Alternative zu Google geworden. Eine der grossen Herausforderungen bei OpenStreetMap (OSM) ist deren schwer bestimmbare Qualität, insbesondere die Vollständigkeit dieses Projekts - auch wenn vielfältige Anwendungen belegen, dass die OSM-Daten "fit-for-use" sind. Mittels modernen Machine Learning (ML)-Methoden ist es möglich, die erwartete Anzahl von Ziel-Features einer Region (Kachel) zu schätzen und zwar durch Zusammenhänge zwischen dem Vorkommen bestimmter "Zeiger"-Objekte. Zum Beispiel deutet die Existenz eines Supermarkts darauf hin, dass auch Restaurants in der Nähe sein sollten.

Aufgabe

Diese Arbeit versucht die Vollständigkeit von OSM-Daten (d.h. Objektklassen) zu schätzen und auf linguistische Variablen abzubilden (beispielsweise "schlecht, mittel, gut"). Die Versuche werden auf urbane D-A-CH-Regionen eingeschränkt. Dabei wird die erwartete ("gelernte") Anzahl von OSM-Objekten intrinsisch - d.h. mittels anderer OSM-Objekte - mit Hilfe von maschinellem Lernen geschätzt und zwar in einer regelmässigen Datenstruktur (zwischen 100 und 500 Metern). Das Verfahren hat das Potential, weltweit und über die aktuell ausgewählten Points-of-Interests hinaus auf weitere Objektklassen (Tags) angewendet zu werden. Diese Arbeit basiert auf bestehendem Projekt, so dass man direkt mit ML-Lernen einsteigen kann, während parallel dazu der Stand der Technik und die Theorie aufgearbeitet wird.

Vorgaben – Technologien – Rahmenbedingungen

- Framework in Python mit Pytorch/fastai, Jupyter Notebook
- Umfeld: Jupyter Notebooks mit nbdev (Entwicklung), fastdoc (Dokumentation), eventuell ReviewNB (git-Support bei Jupyter-Notebooks für Merge-Requests resp. Konflikt-Handling)
- Karten und Darstellungen mit Hilfe von plotly (Graph-Tool) und Folium
- Wahl zwischen Pytorch (fastai?), Tensorflow (Keras?)
- Big Data, Karten-/Geodaten ("Kacheln", GeoJSON)

Vorgehensskizze

- Aufgaben: Einarbeiten. Umgebung aufsetzen. Experimentieren mit Schweizer Strassen, Restaurants, Supermärkte, Schuhläden, etc. Theorie: Verschiedene Ansätze und Probleme.
- Ergänzen des existierenden Projektes "osmcross". Neuschreiben oder Weiter-Entwickeln (Verbesserungen müssen ersichtlich sein).

Termine und Bewertungsschema

Es gelten die üblichen Termine und Regelungen zum Ablauf und zur Bewertung der Arbeit (sechs Aspekte) des Studiengangs Informatik der OST. Zusätzlich wird besonderes Gewicht gelegt auf moderne Softwareentwicklung (Versioning, Continuous Testing & Integration, etc.).

Beteiligte

Diplomanden: Marco Crisafulli and Dominic Monzón

Betreuer: Prof. Stefan Keller, Institut für Software OST sowie Nicola Jordan, Assistent IFS

Dokument: Aufgabenstellung_BA_FS21_Crisafulli_Monzon_v1.docx

Management Summary

The community-built geographical data source OpenStreetMap (OSM) gains much traction lately. As it is free to use for any purpose, the question about its quality arises from critics and the community alike. Besides consistency and different types of accuracy, one important quality is completeness. It is defined as the presence or absence of entities, individual attributes, or relationships. There is currently no general solution to determine completeness. Instead, two approaches are described in the OSM wiki. External data where the present entities are compared to an external dataset. Whereas in internal data, a well-mapped area is used to compare it to another and derive insight from this comparison. This work is based on the latter and envisions to lay the groundwork for an open-source, community-driven, and general solution to the completeness problem. Besides the vision, this work is constrained to estimating points of interest (POIs) in urban areas of Switzerland and neighboring countries. The use outside of this restriction is briefly addressed.

Some of the implemented concepts and ideas in this work come from the predecessor work OSMCross. It is not pursued further and therefore no longer usable in its full functionality. There is other similar work in research that takes on the quality problems with OSM. A majority focuses on positional accuracy and not on completeness. One evaluated work by Kaur and Singh, 2018 also uses machine learning on intrinsic data with a focus on completeness. However, more on error correction and identifying missing attributes on existing entities. Another work by Fritz et al., 2021 is using a regression model to estimate the completeness of bus lines on a global scale. Even if it is a small field, the academy is interested in this kind of work.

The amount of data and the nature of the problem is well suited for machine learning methods.

The first naïve approach that should have served as a base model was trained with high resolutions imagery. Each image was split into 100m x 100m segments. For each segment, the number of POIs was counted. So, the images are the predictor variable, and the number of POIs the continuous outcome variable in a regression problem. As machine learning framework fast.ai is used. It is a high-level wrapper around pytorch, currently one of the most used deep learning frameworks. The wrapper is easier to learn and allows faster progress in an experimental project like this. Thanks to fast.ai and a technique called transfer learning it was possible to retrain an existing machine learning model that already performs a similar task. Surprisingly the model performed well and predicted indeed more POIs in areas where there actually are more POIs. The downside of this approach is the missing high-resolution imagery outside Switzerland and that it only works for POIs in general. Estimating a single category of POIs does not work at all. Therefore, this approach was not pursued further.

The subsequent trained models all rely on intrinsic data. So, no other data than what is present in OSM or derived from it has been used. The first step is to extract the OSM data and load it into a format usable in python. The buildings need to be converted into points, many data can be ignored, and some useful distinctions into categories were made. Additionally, further data

derived from OSM is calculated, and the map is split into tiles of the same size. This was built as a reusable and expandable data preparation pipeline in python. The first relation between POIs explored was the count of POIs per category in a tile. The idea behind this is the following assumption: in a tile where many gastronomy POIs are counted, the probability of finding shops is higher. As there are tiles with no or little data, other relations like the distance to the k nearest POIs of a category were calculated.

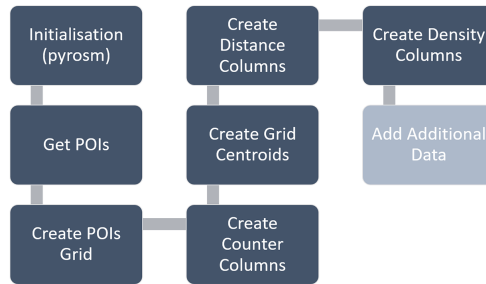


FIGURE 1: Schema of the Data Preparation Pipeline

After experimenting with different models, metrics, and input features, the final model could be optimized enough to show a clear trend in which area the focus of further mapping should lay. On a validation dataset, around 67% of the relevant cells are predicted +/- 1 correctly. Relevant in this case means cells that are not without POIs like water, forest, or similar. With a tolerance of +/- 3, this value is already about 84%. So even if it is not precise in the sense that precisely one shop is missing in exactly this cell. It is precise and accurate enough to decide where resources should be spent to investigate regions further. A traffic-light-like color coding was implemented to make the model more understandable and interpretable. Where green/white means complete, orange means improvable, and red denotes incomplete. Also, a demo was set up to make this at least partially interactive. It is currently not feasible to predict areas on the fly as the data preparation computation is too demanding. Nevertheless, there are precalculated areas that can be explored.

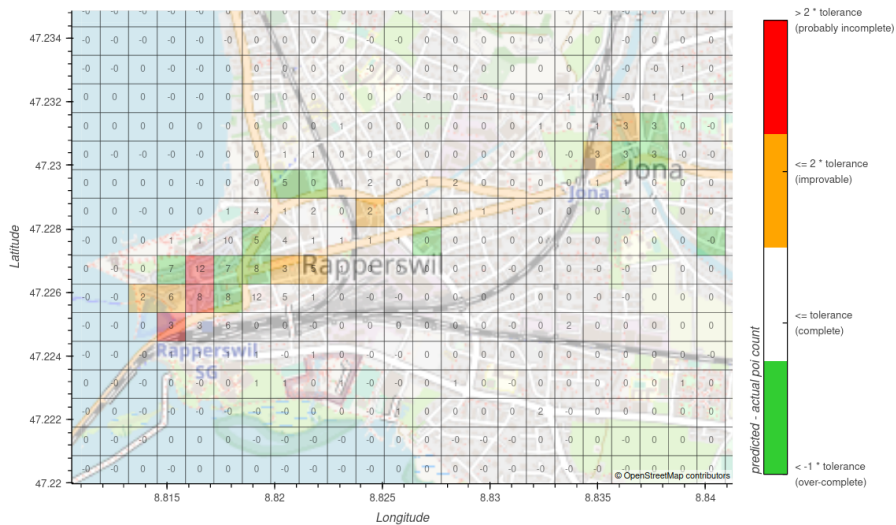


FIGURE 2: Estimated Completeness of Shops for Rapperswil

The work can be improved regarding computation efficiency. If the computation time for data preparation and inference can be reduced substantially, it should be possible to predict areas on the fly. Currently, the prerequisites, the installation, and the usage are too technical for a wide adoption by users unfamiliar with the field. To avoid the technical hurdle, a service-based approach was discussed and set up as a demo. For the demo to become a usable service besides the on-the-fly data preparation, the look, and feel must be optimized, and a proper infrastructure set up. Experiments must be done with different training areas and different validation areas to increase the usefulness further. This should help to build trust and show the usefulness outside Switzerland.

Chapter 1

Technical Report

1.1 Introduction

OpenStreetMap (OSM) is a community-built data source providing geographical data which is free to use for any purpose. Thus the project is attractive for all kinds of applications relying on geographical information.

1.1.1 Problem Definition, Vision

Points of Interest (POIs) have a high market value, and the data provided by OSM has become an equally usable alternative to other map providers such as Google Maps. One of the challenges hindering the widespread use of OSM data is that the quality of the data is hard to determine. ISO 19157 is a standard establishing the principles for describing the quality of geographic data. It defines completeness, logical consistency, positional accuracy, temporal accuracy, and thematic accuracy as the core data quality elements.

The focus of this work is to provide a model that estimates the completeness of OSM POI data based on existing POI data. The estimation is mapped with a traffic light color-coding (red - incomplete, orange - improvable, green - complete) and based on predefined categories. The individual categories are groupings of OSM tags. For example, the gastronomy category is composed of POIs with the tags bar, biergarten, cafe, fast food, food court, ice cream, pub, and restaurant.

1.1.2 Goals

The goal of this work is to provide a machine learning-based approach for completeness estimation of POI categories onto linguistic variables. This will be made available to the general public as an open-source project for further use and improvement. A mockup of how the results could be presented to the end-user is shown in figure 1.1 and displays a color-coded grid on top of a map of Rapperswil.

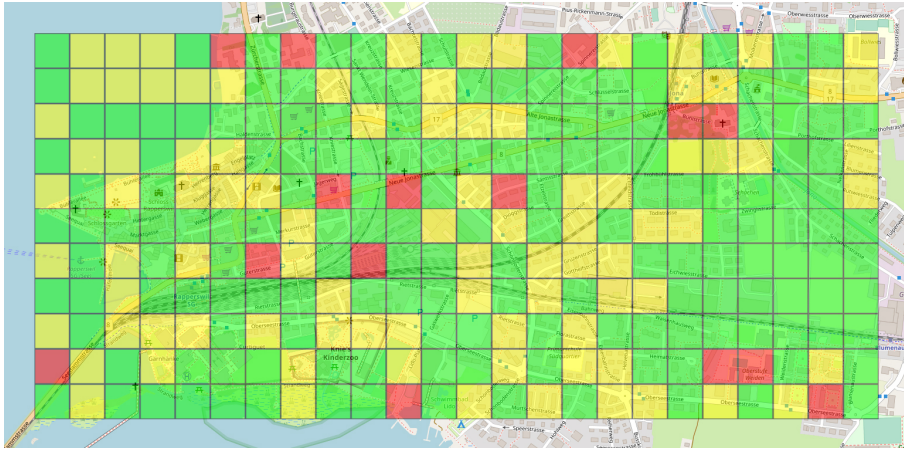


FIGURE 1.1: Mockup of Visual Presentation of Results With 3 Color Categories

Optionally, an interactive web application will be created that uses the resulting model to produce a visual representation of the completeness estimation for predefined areas. This web application serves as an easy-to-use demonstration of the results of this project and as a proof of concept for further implementations.

1.1.3 General Conditions, Definitions, Delimitations

The target region of the model is limited to the central European region of Switzerland and its neighboring countries due to their geographical and demographic similarities. Additionally, only a predetermined subset of features will be estimated. Further, the completeness of POIs should be estimated for regularly sized areas with a grid size between 100 and 500 meters. The methods described in this thesis could be used or adapted to estimate the completeness of additional classes globally. Re-training with a suitable area in a different geographical or demographic region would allow for prediction in the respective region.

1.1.4 Method

The project was carried out in the following four main phases.

Familiarization

In the first phase, the preliminary project was analyzed to understand the methods used for data preparation and prediction. This also facilitated creating a rough plan of the steps to be taken to carry out the work. In parallel, the machine learning know-how needed for the work was developed with the basics from Howard and Gugger, 2020 and by experimenting with OSMCross.

Basic Approach

Next, a custom pipeline for data preparation was implemented to acquire, filter, and process the OSM data necessary to generate training data. A simple model was created which attempted to estimate the number of total POIs in a grid of one-hectare cells only relying on high-resolution aerial images.

Implementation

The findings of the basic approach were used to adapt the existing data preparation pipeline and improve it accordingly. Additionally, the training process was changed to only take intrinsic data as input. These changes allowed for more precise predictions in a broader selection of features. The types of values used as training data were the effective count of POIs of a particular category per cell, the average distance to the three nearest neighbors of the same category, and the density of POIs in a 500m radius around the center of the cell. Figure 1.2 shows a visual simplification of how nearest neighbors and density were calculated.

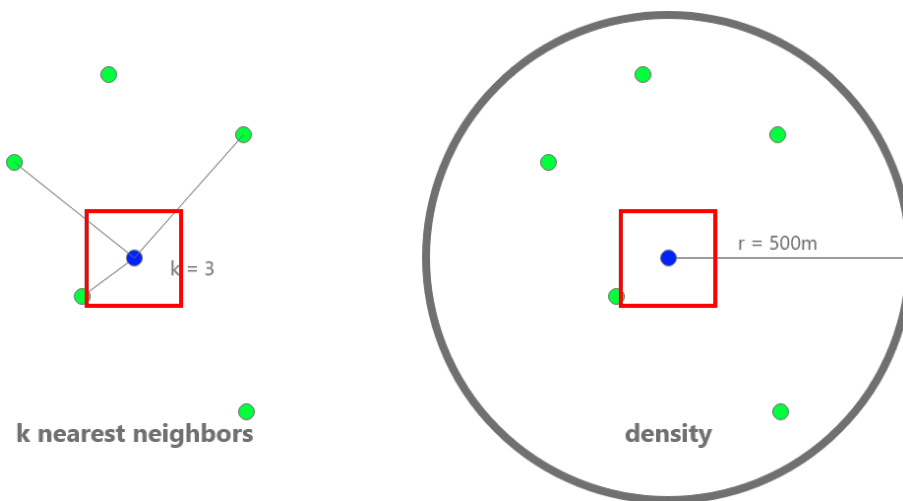


FIGURE 1.2: Visualization of KNN and Density Calculation

Optimization

With the model now in place, various optimizations were made to the model in further iterations. For example, the data preparation was supplemented by additional intrinsic data such as Land Usage and Land Coverage (LULC) and the option to include extrinsic data. Further, the training process was optimized experimentally so that the prediction accuracy could be further improved.

1.2 State of the Technology

In this chapter, various existing standards and solution approaches are reviewed. Additionally, the requirements for the project are described.

1.2.1 Existing Solutions and Standards

ISO 19157 Geographic Information - Data Quality

ISO 19157:2013 Geographic information - Data quality sets out the principles for describing the quality of geographic data by, among other things, defining components for describing data quality.

The core elements to describe quality established through the standard are completeness (commission, omission), logical consistency (conceptual consistency, domain consistency, format consistency, topological consistency), positional accuracy (absolute or external accuracy, relative or internal accuracy, gridded data position accuracy), thematic accuracy (classification correctness, non-quantitative attribute correctness, quantitative attribute accuracy), temporal accuracy (accuracy of a time measurement, temporal consistency, temporal validity), usability (based on user requirements)¹.

General Approaches

Depending on the data they are relying on, existing methods to measure the completeness of OSM data can be divided into subcategories. Methods relying on external data mostly base their estimations on comparing two data sets mapping the same area. In contrast, those relying on internal data only use the information gained from within the OSM data. The OSM wiki lists the following examples for those approaches ("OpenStreetMap Wiki", 2021).

External Data

Numeric comparison Count elements in a given area and compare the amounts to the equivalent counts from external sources.

One-to-one comparison Compare features one by one.

Population based estimation Derive likely number of features based on population data.

Internal Data

Cross locational comparison Well-mapped areas are used to derive metrics for likely numbers of features.

Feature accumulation The addition of new features asymptotically decreases the better an area is mapped.

Feature density Calculating the number of nodes per area can help identifying under-mapped locations.

Existing Solutions

A systematic review of the literature to identify the current focus of research in terms of OpenStreetMap data quality performed by Kaur et al., 2017 revealed that the majority of researchers focus on the positional accuracy of the data and achieve their results by comparing OSM data to other datasets.

¹<https://www.iso.org/standard/32575.html>

An example of such work is a study by Hecht et al., 2013 where building footprints from topographic surveys or digital topographical data of cadastral land registers is compared to with the corresponding OSM data to measure the completeness of building footprints. Figure 1.3 shows an example of the results of such a comparison.

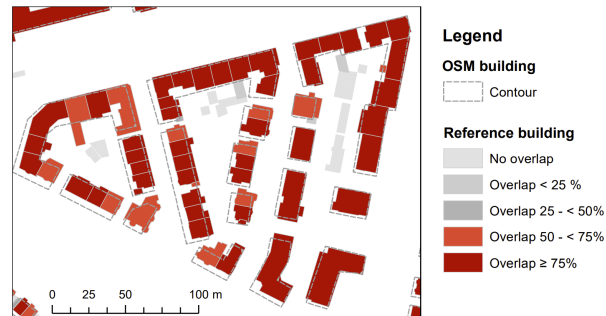


FIGURE 1.3: Overlap Between Official and OSM Datasets
(Hecht et al., 2013)

In contrast, Fritz et al., 2021 take an intrinsic approach to globally estimate the completeness of local bus routes using a regression model.

Another promising intrinsic approach is taken by Barron et al., 2014. Various methods are presented that attempt to make statements about data quality based on the data history. These methods can be applied globally and work without the use of external reference data.

An approach similar to that taken in this project is being taken by Kaur and Singh, 2018. It proposes a machine learning-based solution to improve OSM data quality by using intrinsic data. The main difference is the focus on detecting and correcting errors in existing data. For example, missing or misplaced attributes on existing nodes and ways are detected and rectified.

The preliminary project to this work, OSMCross², pursues, in essence, the same goal as this project, but is not developed further and is therefore no longer usable in its full functionality. An example of a prediction of completeness of gastronomy objects in Bern is shown in figure 1.4.

²<https://ifs.hsr.ch/index.php?id=19673&L=4>

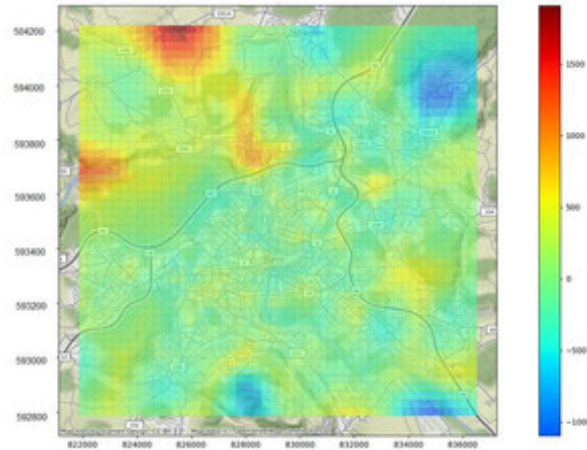


FIGURE 1.4: Completeness of Gastronomy Objects in Bern Estimated by OSMCross
<https://ifs.hsr.ch>

Deficits

External data sources may not be available for public use due to copyrights. To get the most information out of the comparison with external data, one has to assume that the external data set is complete or at least has better quality than the data set it is compared to. Further, additional data like population data might not be available with sufficient precision or outdated, leading to inaccurate results. Additionally, the quality of external data varies by region as some provide more complete data than others.

Internal data, on the other hand, is readily available but has its caveats. Thus, only projections or estimates based on correlations are possible. Identifying these correlations is difficult on the one hand and also dependent on the quality of the data already available on the other.

1.2.2 Use Cases

Actors

System The provided software.

User (Maintainer) Developer who wants to maintain or extend the current functionality. (Example: Student who wants to add income data to the training data.)

User (Tech-User) Technically savvy user who wants to reproduce the results obtained or obtain comparable results with their own data. (Example: Mapper who is interested in a prediction for an area which is outside of the scope of this project.)

User (End-User) Person who wants to see the results of the project but is not interested in adapting or changing the software. (Example: Person who wants to know if a certain category of POIs is completely mapped in his neighborhood.)

Transform Geometries

Actor: System

Geometry data is converted by the system in such a way that no important information is lost. Geometries of the polygon or multipolygon category are converted to point data for further processing using appropriate methods.

Import OSM Data

Actor: User (Maintainer, Tech-User)

The user can import OSM data and prepare it for further use. The OSM data is in pbf format and the user can choose whether to import the complete data set or only a part of it.

Add Additional Data

Actor: User (Maintainer)

The user can extend the data preparation so that additional intrinsic or external data can be read in and used for training.

Predict Selected Area

Actor: User (Maintainer, Tech-User)

The user can reproduce the predictions or make predictions for a custom area.

Presenting Results

Actor: System

The system presents the results in a color-coded grid over a base map. Additional information can be inspected in detail per cell.

View Results

Actor: User (Maintainer, Tech-User, End-User)

The user can view the results for predefined areas.

1.2.3 Non-Functional Requirements

Usability

The project should be documented so that a subsequent set-up of the infrastructure is possible, and the results achieved can be reproduced understandably. The user is supported in the correct execution by appropriate instructions.

Reliability

The software should not crash when executed on sufficiently strong hardware and should output a meaningful error message in the event of an error. The user should be prevented from executing the sequence incorrectly.

Performance

There are no explicit performance requirements. The execution times of the preliminary project are used as a guideline.

Supportability

Since the work is to be made available to a broad audience and further development should be possible, the project must be structured and documented so that it is easy to add additional functionality and extend it to other areas.

1.3 Evaluation

1.3.1 Development Environment and Libraries

Jupyter Notebooks were chosen as IDE early in the project. Aside from being a de facto standard in data-science projects, they provide the ability to display code, plots, and documentation all in one place. To overcome some shortcomings of Jupyter Notebooks, namely the creation of libraries, testing, ease of refactoring, and CI/CD, nbdev by the creators of fast.ai was used. Some additional libraries like Pandas and its extension GeoPandas were chosen without further evaluation since those are de facto standards in the data science community.

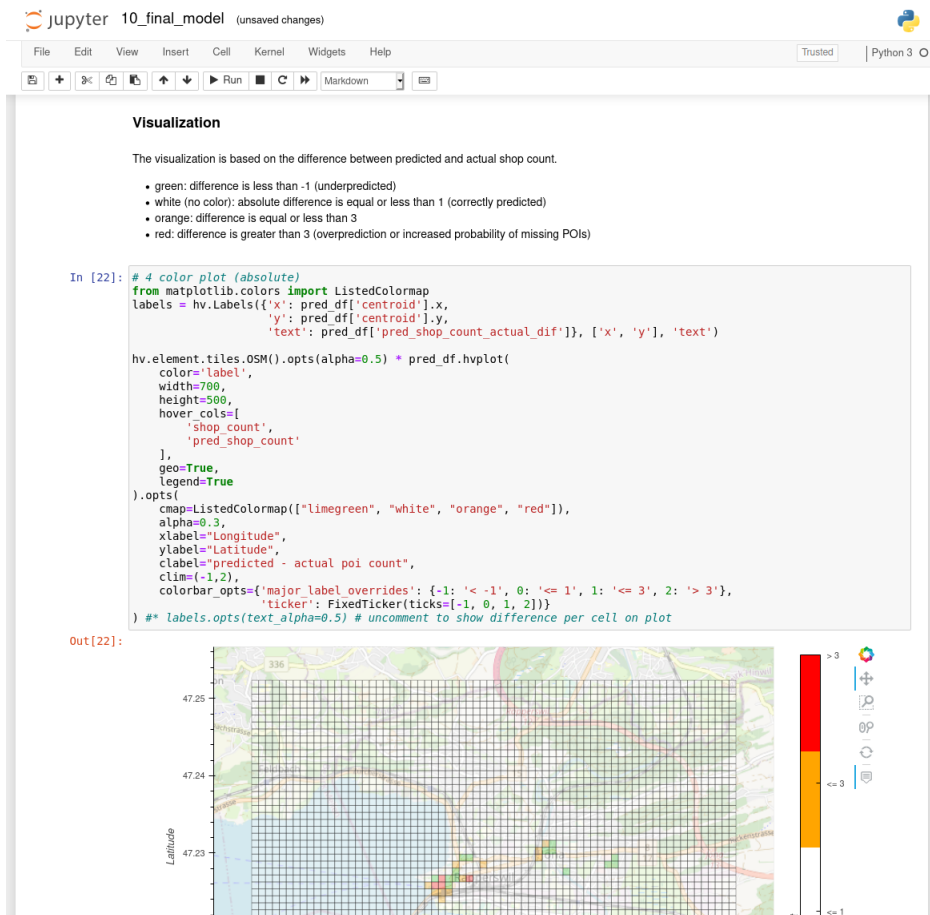


FIGURE 1.5: Jupyter notebook screenshot

OSM Data Extraction

OSM data is generally made available in Protocolbuffer Binary Format (PBF) format by the various providers. To read those files into a datatype that can be used, a library like pyrosm or OSMnx is needed. Pyrosm was chosen because it was one of the most recommended and the preliminary project also relied on it. The possibility to define custom filters provided by pyrosm allowed to pre-filter the data ahead of further analysis. That way, only the necessary data could be loaded into the data preparation pipeline instead of the whole dataset.

1.3.2 Machine Learning Framework

TensorFlow with Keras

TensorFlow together with Keras, is an end-to-end open-source platform for machine learning developed by Google.

Table 1.1 shows the pros and cons of TensorFlow with Keras.

TABLE 1.1: Evaluation TensorFlow with Keras

Pros	Cons
Detailed, up-to-date documentation	Slower than other frameworks
TensorBoard visualization toolkit	Harder to learn
Large community	
Production deployability	

PyTorch with fast.ai

PyTorch is an open-source machine learning framework. The deep learning library fast.ai provides high-level components that allow rapid prototyping and low-level components which can be combined to build new approaches.

Table 1.2 shows the pros and cons of PyTorch with fast.ai.

TABLE 1.2: Evaluation PyTorch with fast.ai

Pros	Cons
Modular	Partially incomplete documentation
Pretrained models	Production deployability
Easy to write custom models	
Transparent model behaviour	
Pythonic	
Supports TensorBoard	

Decision

Both frameworks are suitable for the implementation of the project. In the latest versions, both allow a quick and relatively simple start and offer possibilities for further development.

Since PyTorch with fast.ai is currently preferred in research since even complex architectures can be built relatively quickly. Thus fast development of prototypes is possible, PyTorch with fast.ai is used for this project.

Plotting

There is a large number of different plotting libraries and tools. As all have a similar set of base functions, they were evaluated on the criteria of ease of use, ability to print geo data, interactivity, solid user base if support is needed. The tools taken into consideration were matplotlib, plotly, seaborn, and hvplot.

matplotlib One of the most used plotting libraries in python. Not the simplest to use, but it has a large number of available plots. It is not interactive, and it is not designed to work with maps.

plotly Has gained much traction lately and has a good amount of beautiful-looking plots, including maps. One of the favorite libraries in data science applications.

seaborn It is based on matplotlib and provides a high-level interface that provides a sane default for good-looking plots. Smaller userbase than the other but capable of drawing maps.

hvplot A high-level plotting library for the PyData ecosystem. It has a large userbase and is capable of plotting maps. It has a steeper learning curve than the others.

As some libraries of the PyData stack were already chosen (Pandas and GeoPandas), the choice was made to use hvplot.

Boosting

XGBoost is one of the most used libraries and was chosen since it is supported by fast.ai.

1.3.3 Demo

The notebooks can be a valuable tool for data exploration and experimentation, but from a demo perspective, they are not quite perfect. Anyone that wants to use them needs some knowledge about python and Jupyter Notebooks. Further, to run this project, some docker knowledge is necessary as well. The installation is rather complicated for someone who is not familiar with the used tools. The best case would be a website with an elementary set of features that shows the critical insights of the work. Two solutions that promised precisely this were evaluated. Streamlit and Panel.

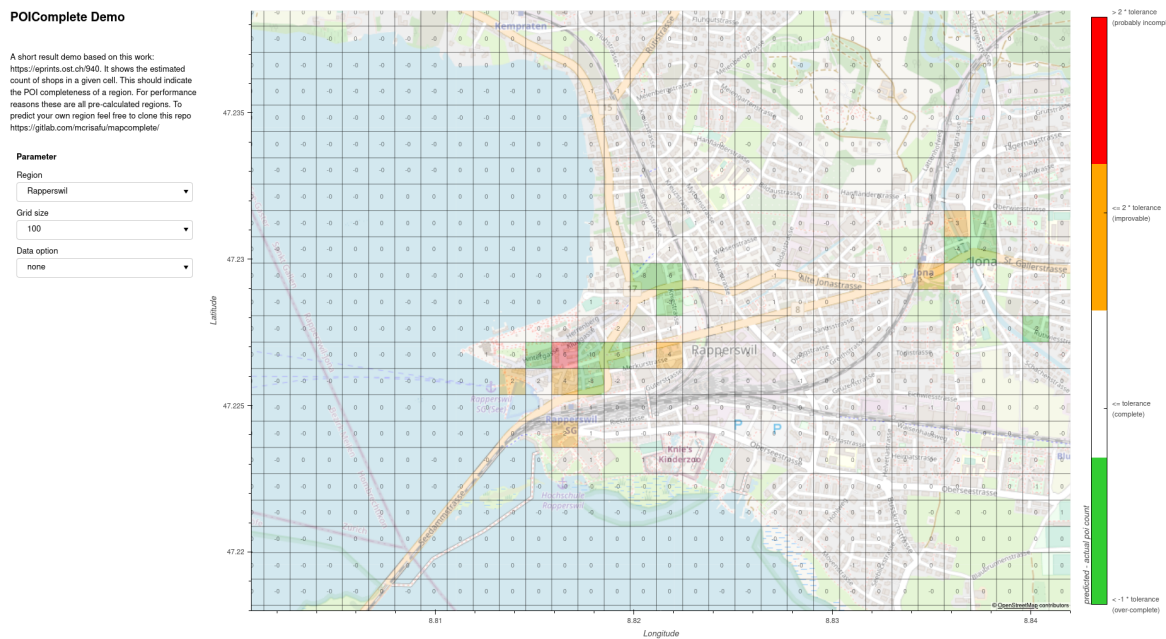


FIGURE 1.6: Screenshot of Demo Optimized for FullHD (1920x1080px) Screens

Streamlit

Streamlit³ claims to be the fastest way to build and share data apps. It is an open-source python library with good documentation, a large community (~15k Github stars), and a simple API. The installation worked flawlessly, and the demo app looked promising. The problems began when reusing plots already defined in the notebooks. One of the requirements for this demo application was to reuse as much code, plots, and libraries as possible. So the out of the box experience was sobering. There were problems with multiple libraries, and after some research, they seemed not so simple to solve. As this happened early in the evaluation, the approach was discarded.

Panel

The other library tested was Panel⁴ from Holoviz. It describes itself as a high-level app and dashboarding solution for python. It is also open source but has a significantly smaller community (~1k Github stars), and the documentation looks useable but promises a steeper learning curve than streamlit. For these reasons, the library was only the second choice. The advantages of the project certainly are the tight integration with Jupyter Notebooks and the already used plotting ecosystem as it is from the same developers.

Panel can serve applications defined in plain python files or Jupyter Notebooks. For simplicity of the deployment, the demo should be a docker container. So, the plain python file option was chosen as it reduces the complexity of the container. In retrospect, this was not the best choice.

³<https://streamlit.io>

⁴<https://panel.holoviz.org/>

The learning curve was steep, but at least it worked out of the box as it should. The complications began with the size of the data. The generation of the plots sometimes took so long that panel ran into timeout errors. These timeouts are set to a generous 900 seconds. Further analysis could not clarify why the generation of plots takes substantially longer in panel than in the notebook. For comparison, the generation of the same plot, which ended in a timeout in panel, took 11 seconds to plot directly in the notebook.

Fortunately, Holoviz is aware of the problem that large datasets take much time to plot and created a library to speed things up. Datashader⁵ is a graphics pipeline system for creating meaningful representations of large datasets quickly and flexibly.

The basic idea is to convert the data to an image server-side and send back the image instead of the whole plot object, containing all the data. With Datashader it is possible to calculate these images on a GPU with CUDA or even on a computation cluster with Dask. In theory, a vast amount of data can be visualized with this library. Luckily this was not necessary for this project. After some experimentation with Datashader the plot time in the notebook could be reduced even more. However, this did not affect the plain python panel demo.

To verify that the cause of the problem is the plain python approach, a demo Jupyter Notebook was created. Surprisingly this worked a lot better. Consequently, the plain python approach was discarded.

The higher demand in storage capacity necessary for docker images with Jupyter is a negligible trade-off, considering the low prices. The possibility to directly document in the notebooks made this approach even more desirable.

1.4 Implementation Concept

1.4.1 Base Model Using Orthophotos

The initial approach to determining the completeness of POI data was to estimate the number of POIs using only aerial photographs. Aerial photographs from Swissimage served as the data basis. Swissimage provides high-resolution images of the whole of Switzerland with a ground resolution of up to 10cm as GeoTIFF in a 1km grid.

A 10km x 10km grid with the center in the city of Zurich was chosen as the training area to make a forecast for Rapperswil and the surrounding region. For easier handling, the one km² tiles were divided into 100x100m tiles. Thus, a total of 10'000 individual images with the corresponding number of POIs per tile from OSM were available as training data.

⁵<https://datashader.org/>

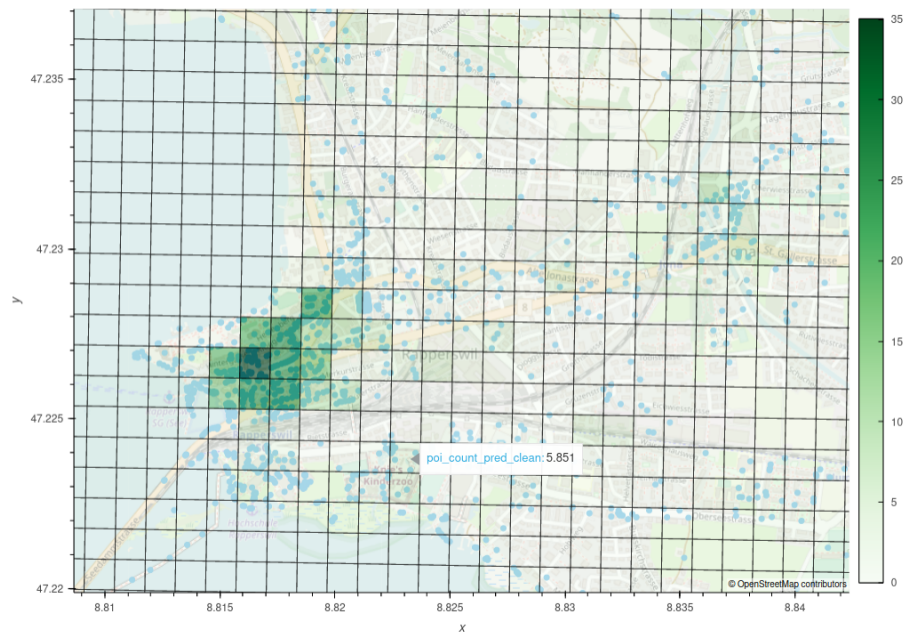


FIGURE 1.7: Prediction of Number of POIs for Rapperswil Based on Aerial Images

The use of a pre-trained model, here Resnet34, allowed to easily further train a model that has already been trained with a large dataset. Based on this model, a model for predicting the number of POIs based on aerial photographs could be trained employing transfer learning.

Findings

- Areas covered by water normally do not have POIs and can be recognized easily; thus, these areas already get accurate predictions.
- Even though no further information was used to train the model, the center of Rapperswil is correctly predicted with the highest number of POIs.
- As the POIs are not further filtered, the model is trained with data that is unsuitable for visual predictions.
- The model generates somehow useful results for POIs in general while deteriorating rapidly if only a certain POI class should be predicted. It is comprehensible that aerial images alone do not provide sufficient information to differentiate between a shop and a restaurant.

1.4.2 Data Preparation Pipeline

The raw data and the base for this project are PBF files that can be downloaded from providers like Geofabrik⁶. For a consistent, repeatable and expandable data preparation pipeline the `prepare_data`-module was created. The pipeline consists of the following single steps (figure 1.8) all working in sequence:

⁶<https://download.geofabrik.de/>

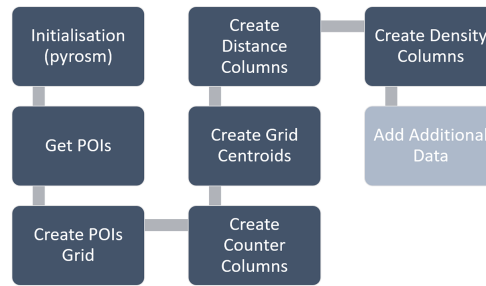


FIGURE 1.8: Schema of the Data Preparation Pipeline

Initialization

In the initialization phase the PBF gets read with `pyrosm`. Additional filters like the `pyrosm_filter` (filters OSM data to specific types or tags) or the `feature_filter` (filters tags further and groups them together) get initialized.

Get POIs

This step extracts all elements defined in the `pyrosm_filter` out of the OSM data in the PBF. It also projects them into WGS84/Pseudo-Mercator (EPSG:3857). This is needed as the next step, getting the centroid, is somewhat imprecise in a non-metric projection. Since only `POINT` objects can be used in further calculations, the other types of objects, namely `POLYGON` and `MULTIPOLYGON` need to be converted to `POINTS`. This step is essential as a substantial amount of the provided OSM data would be ignored if only `POINT` data would be extracted. Thus the centroids of the objects are calculated, resulting in a homogenous structure of `POINTS`.

Create POIs Grid

For most tasks like computation of additional data, training, and prediction, a grid over the POIs is needed. The max and min coordinates of the POIs are extracted, and with the help of `numpy.arange` an evenly spaced grid is computed. The `step` argument is the grid size in meters converted to a haversine distance. For simplicity reasons, the distance calculation is only an approximation and most precise around 47°N and 8°E.

Create Counter Columns

The first set of data calculated for training is the count per tag group per grid cell. The group is defined in `feature_filter`. It simply counts how many POIs of a defined group are found in the cell.

Create Grid Centroids

Some of the calculations in the following pipeline steps require the centroids of the grid cells. For the same precision reasons as in step *Get POIs*, the grid is projected to WGS84/Pseudo-Mercator (EPSG:3857).

Create Distances Columns

Similarly as in OSMCross the proximity p measure also gets calculated. It is defined as the sum of the haversine distance d to the k nearest neighbors divided by k .

$$p = \frac{\sum_{i=1}^k d_i}{k} \quad (1.1)$$

Create Density Columns

Also, as in OSMCross, the density per cell gets calculated. It is defined as the count of POIs per group in a given radius around the cell centroid. This is achieved with the `geopandas.buffer` function.

1.4.3 Extending Data Preparation

One of the advantages of this modular approach for data preparation is the simple and straightforward extension possibilities. As the resulting data is an attribute of the instantiated `DataPreparator` class, all that is needed is an additional class method called in `prepare_data()`. To be usable for training, the additional data must fit in the grid.

1.4.4 Base Model Using Only OpenStreetMap Data

The next model used an intrinsic approach, using only data from OpenStreetMap and other information generated from it. In order to obtain a more fine-grained prediction about individual categories of POIs, POIs were divided into the categories of gastronomy, education, healthcare, public service, transportation, finance, entertainment, waste management, public transport, tourism, art, and buildings based on their tags.

As data for training, the number of POIs in a 100x100m grid was calculated for each category. To enrich the model with further information about the relations of the POIs, two additional parameters were calculated.

On the one hand, the average distance from the center to the three nearest neighbors was calculated for each tile. Furthermore, a density was defined and calculated as the number of POIs of one category within a radius of 500m around the center of a tile.

The region around the city of Zurich was again used as the training area, and the region around Rapperswil was used for the prediction.

Findings

Without further optimization, the predicted number of POIs was already promising. The predictions were generally lower than the actual values, but regional amassments such as in the city center of Rapperswil were visible. On the other hand, the effective values were far off in almost all tiles where multiple POIs were present. Similar to the model with aerial imagery, POIs were correctly not predicted in areas with bodies of water.

1.4.5 Improved Models

Various approaches have been taken to improve the model further.

TABLE 1.3: Existing Experiment Trackers

Attribute	Weights & Biases	Neptune.ai	MLFlow
Open Source	Client only	Client only	Yes
Features	3/5	3/5	4/5
Documentation	4/5	4/5	3/5
SaaS-Cost	\$70/mo	\$49/mo	No SaaS
Ease of use	3/5	3/5	2/5
Self-Hosted	yes	\$900/y	yes

In the first step, further data were added.

Land Use and Land Coverage

To gain deeper insight into how land is used in a tile, Land Use and Land Coverage data were analyzed, and primary land use per tile was added as a category to the dataset. The assumption was that certain forms of land use indicate that little to no POIs (e.g., water bodies) or a higher number of POIs (e.g., retail) can be expected in an area.

Population Data

Assuming that the number of inhabitants in an area also correlates strongly with the number of POIs, population data from an external source, the Federal Statistical Office, was integrated. The FSO provides relatively current (population figures as of 2019) and precise (inhabitants in a 100x100m grid) data. However, the use of this data had several severe drawbacks. On the one hand, the data is not available in this timeliness and accuracy for all regions outside of Switzerland (e.g., in Germany, the most recent data is from 2011). Furthermore, the data are inaccurate for low population densities, as population figures of 1-3 persons are summarized for data protection reasons. Finally, it also contradicts the initial approach of the work to use only intrinsic data for the prediction.

Going forward, further improvements and optimizations to the training process were made.

Machine Learning Optimization

To improve the results, settings for the metric used, optimization function, number of epochs, and batch size were automatically tested and subsequently evaluated.

1.4.6 Experiment Tracker

One of the problems in a highly experimental project like this is how to keep track of experiments. A tool is needed to record the parameters used and the results for each round of experimentation. As this project is not uniquely confronted with this problem, a short investigation over existing solutions was done.

The features that MLFlow offers are way above what is needed in the scope of this project, and the integrations with fast.ai and XGBoost are in an experimental state. Neptune.ai does not offer self-hosting, so Weights & Biases was tested. The setup took some time and is unfortunately not that straight forward as one would hope. However, after the setup, the basic functionality

worked well. For the limited amount of functionality needed in this project, the setup and manual configuration steps are too laborious. Also, it is relatively heavy on the resources. The `wandb/local` docker image is about 1.8 GB big.

Nonetheless, the work with Weights and Biases helped to choose some parts that could be used in a way that works better with the project. So a simple experiment tracker was implemented from scratch. The idea was the same as in all existing trackers. Hook into the callback functions of XGBoost and `fast.ai` and save the learning progress and the metadata to some location.

As it is simple and well understood in all the tools used in this project, two CSV files were used. One for metadata and one for the results. From these CSVs, plots similar to Weights & Biases could be generated with the already familiar plotting library `hvplot`.

For bigger projects or larger teams, this approach for sure does not work, but for this size, it seems a good fit without introducing too many dependencies and complexity.

1.4.7 Blurring

The results showed a pretty good match in a region about 3x3 grid cells of size 100m. So the sum over these nine cells was in some regions a better match than the nine cells alone. Therefore the idea of blurring the cells together was explored. The algorithm is simple. Find all neighboring cells. Calculate the average over these cells.

Unfortunately, as seen in the following plot (figure 1.9), the precision could not be significantly boosted. This method indeed helps to lower the error of a single cell, but as a tradeoff, it introduces an error to more previously correctly predicted cells. The intensity of the color denotes the size of the prediction error. Red is overestimated and blue is underestimated.

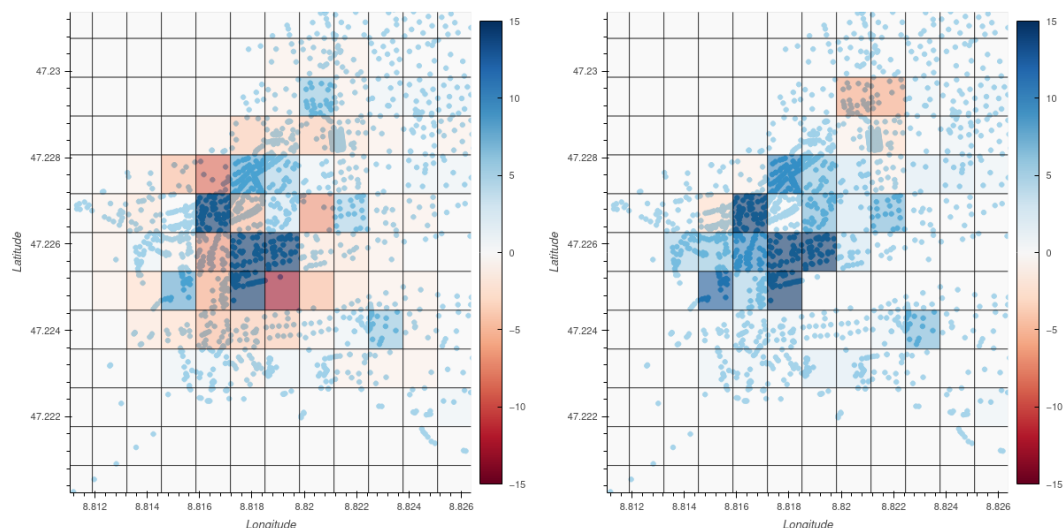


FIGURE 1.9: Left: Blurred Results, Right: Unblurred Results

1.5 Results

The model developed in this thesis allows the prediction of different features by simply adjusting the input parameters for the training. The results discussed in this chapter concern the prediction of shops⁷ and serve as a representative example.

1.5.1 Base Model

Training with fast.ai's default settings already yielded promising results in a visual inspection of the results. To evaluate the results qualitatively as well, four additional metrics were defined. The percentage of overall correctly predicted cells gives information about how good the model is for the prediction on the whole prediction area. Since outside the larger cities a high number of cells without POIs can be assumed to be correctly predicted, e.g. areas with water bodies and a high proportion of non-commercial land, this value can be expected to reach a high value in most cases. The proportion of correctly predicted cells in which at least 1 POI of the investigated category is expected provides information on how many of the cells in which a POI is present are predicted accurately. The proportion of correctly predicted cells with a tolerance of 1, or 3 respectively, in which at least 1 POI of the category under investigation is expected provides information about how many cells are predicted with sufficient accuracy. Depending on the requirements and cell size, this tolerance can be increased further.

This results in the following values for a prediction of the number of shops in the Rapperswil training area.

- Correctly predicted: 96.815%
- Correctly predicted (not 0): 10.000%
- Correctly predicted (not 0, tolerance +/- 1): 54.286%
- Correctly predicted (not 0, tolerance +/- 3): 67.143%

A closer look at the results (figure 1.10) shows that the differences between predicted and actually recorded value tend to be underestimated, but the deviation reaches larger values when overestimated.

⁷https://wiki.openstreetmap.org/wiki/Map_features#Shop

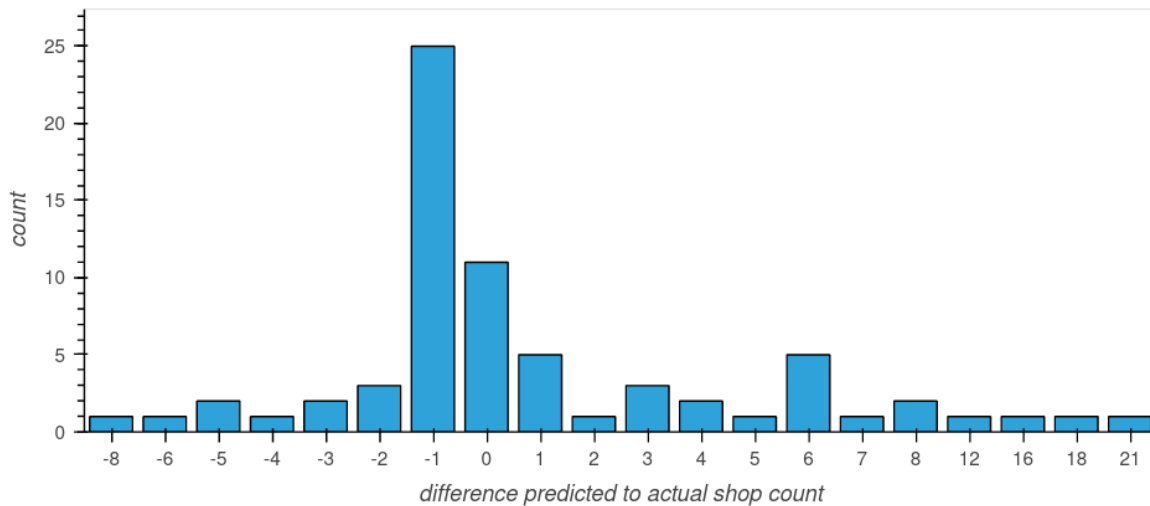


FIGURE 1.10: Difference Between Predicted and Actual Shop Count for Base Model

1.5.2 Experiments to Improve Results

Number of Epochs

When comparing the results for training and validation loss, it becomes apparent that training loses effectiveness from epoch 5 onwards (curve flattens) and from epoch 10 onwards there is hardly any improvement, but the training and validation loss diverge more and more from each other. This implies that the model is overfitting, which in turn means that the prediction on the training data might improve, but the general application on other data is no longer possible. Thus, the training is terminated after 10 epochs. Figure 1.11 shows the course of the training and validation loss over 25 epochs for all conducted experiments.

Tested number of epochs 1 - 25

Selected number of epochs 10

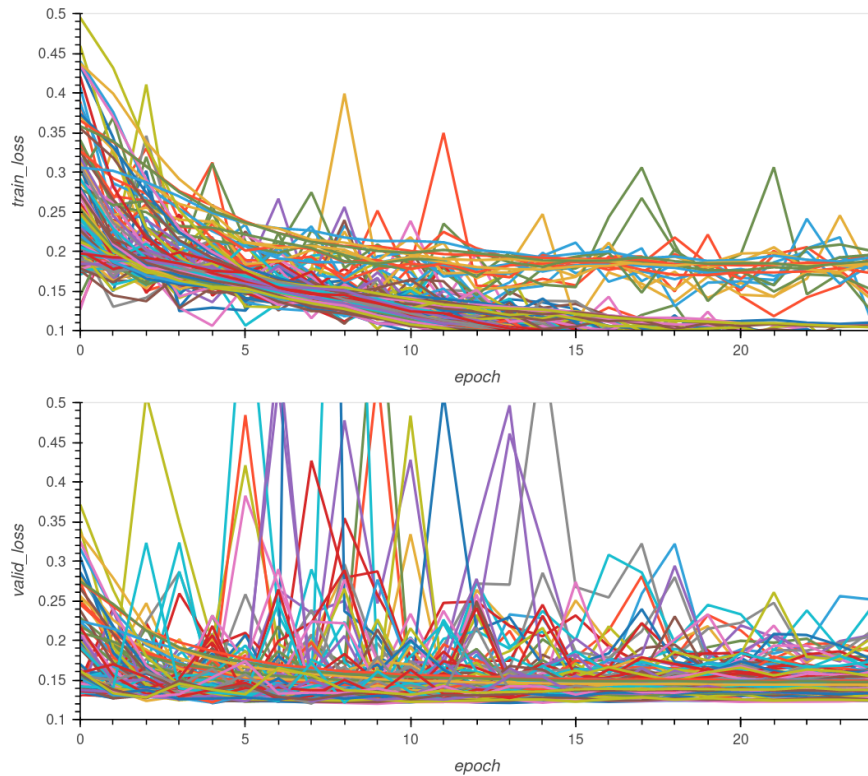


FIGURE 1.11: Training and Validation Loss by Experiment

Metrics

Figure 1.12 shows that the best metric depends on the optimization function chosen. For example, the mean absolute error gives good values without large outliers for Adam and Lamb. For the remaining optimization functions, there are also only comparatively small outliers. The remaining metrics have the problem that although individual good values are obtained, significant outliers still occur even for many epochs. The exception here is SGD, for which there are few to no outliers, but the results are only in the lower midfield.

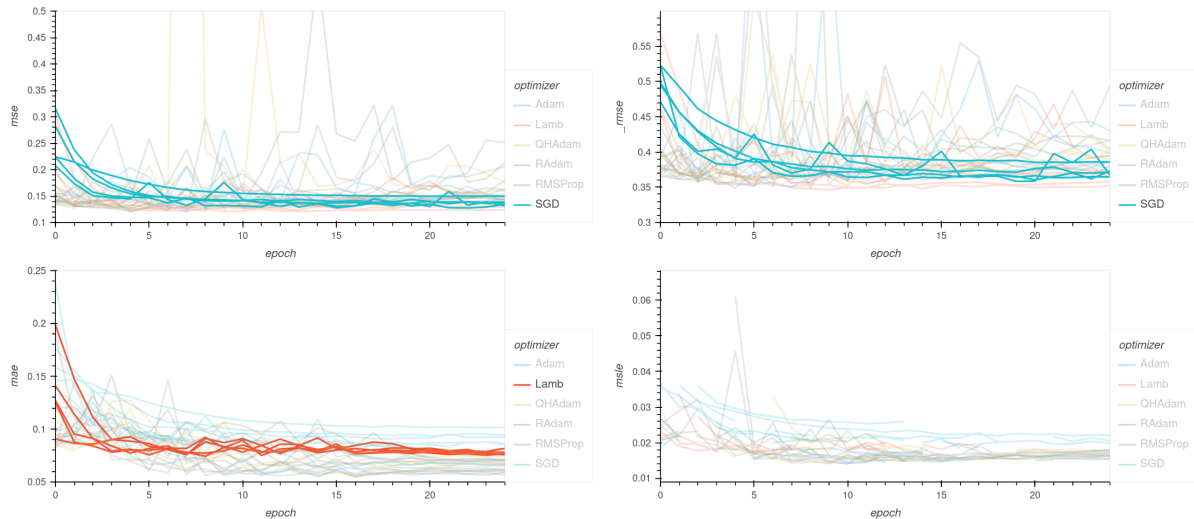


FIGURE 1.12: Metrics by Optimizer

Since the values for the mean absolute value approach the best possible value quickly and reliably across all optimization functions, this metric was chosen.

Tested metrics Mean Absolute Error (MAE), Mean Squared Error (MSE), Mean Squared Logarithmic Error (MSLE), Root Mean Squared Error (RMSE) ⁸

Selected metric MAE

Optimizer Functions

The choice of the optimization function was based on the criteria difference between training and validation loss and effective validation loss. The difference between training and validation loss indicates how strong the models is overfitting or underfitting. Ideally, the difference should be close to zero, as this would implicate that the model delivers similarly good results for the validation data as for the training data. On the other hand effective validation loss indicates how good the prediction on the validation dataset is.

⁸<https://docs.fast.ai/metrics.html>

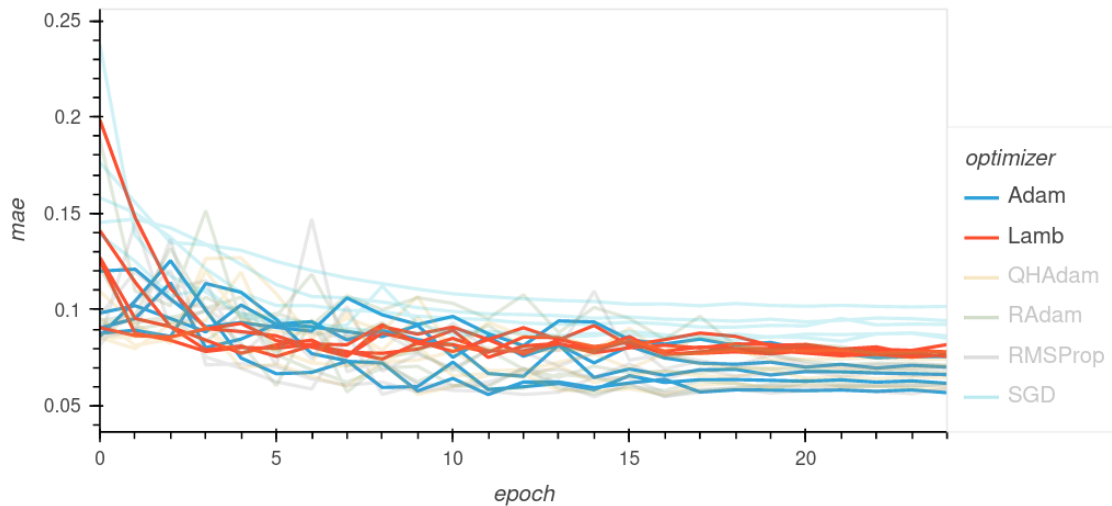


FIGURE 1.13: Mean Absolute Error by Optimizer (Adam and Lamb in Focus)

As shown in Figure 1.13, all tested optimization functions demonstrate comparable results in terms of mean absolute error. However, the functions Adam and Lamb have the advantage that they reach low values on the y-axis on the one hand and on the other hand, they reach a good value already early (after 5 epochs) without having large outliers. QHAdam, RAdam and RMSProp finally reach comparable values as the other functions, but have partly strong outliers especially in the early epochs. Since the training should be repeatable for other regions, optimizers without outliers are preferred and the aforementioned optimizer functions are therefore dropped.

Tested optimizer functions Adam, Lamb, QHAdam, RAdam, RMSProp, SGD ⁹

Selected optimizer functions Adam, Lamb

Batch Sizes

From the results with different batch sizes, it can be seen that this has a large influence on the result, especially with a small number of epochs. For the Lamb optimization function and with the mean absolute error as a metric (Figure 1.14), it can be deduced that the graphs are almost congruent from epoch 4 onwards.

⁹<https://docs.fast.ai/optimizer.html>

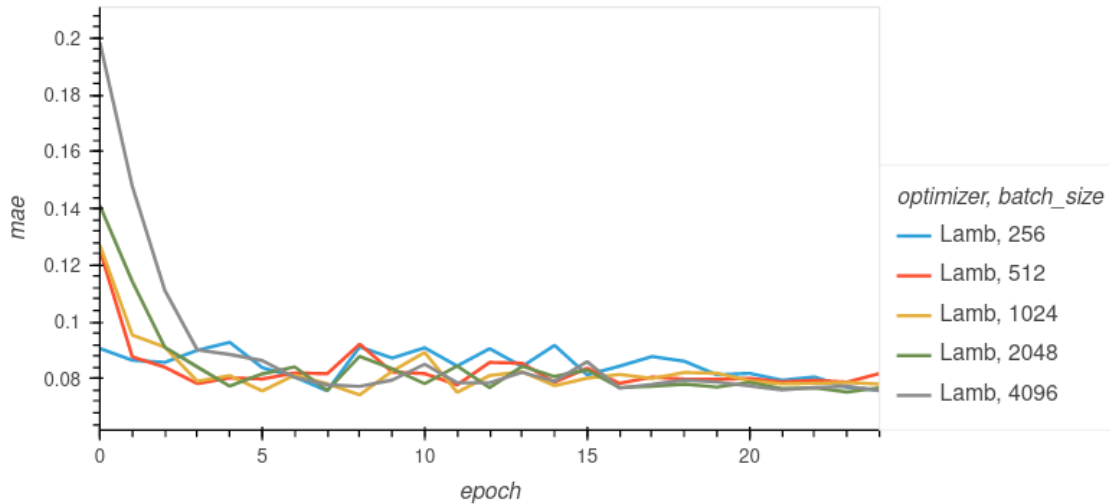


FIGURE 1.14: Mean Absolute Error by Batch Size for LAMB

On the other hand, the choice of batch size can also have a significant influence on the result, as is the case with the Adam optimization function and the mean absolute error metric. In the first epochs, as in the first example, it is evident that a larger batch size leads to comparably good values being calculated only in later epochs. However, with a higher number of epochs, the values do not converge as strongly as with Lamb, but run quasi parallel to each other (Figure 1.15).

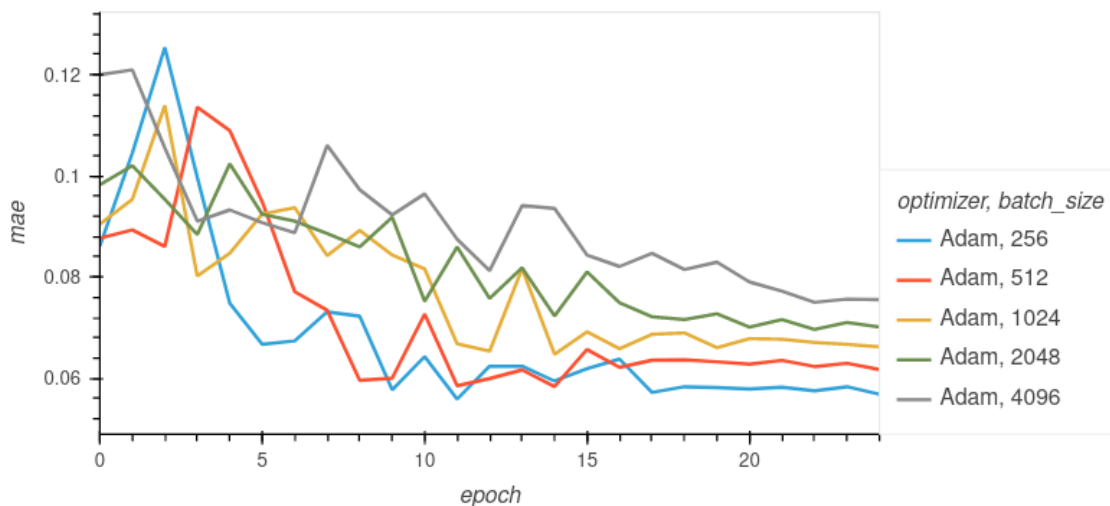


FIGURE 1.15: Mean Absolute Error by Batch Size for Adam

As it yields the best results with Adam a batch size of 256 was chosen for this optimizer function. Since training times are significantly faster the larger the batch size is and the results do not differ significantly from each other, a batch size of 4096 is chosen for Lamb.

Tested batch sizes 256, 512, 1024, 2048, 4096

Selected batch sizes Adam: 256, Lamb: 4096

1.5.3 Improved Model Quality

As in the previous experiments, the model with the optimized values was applied for a prediction in Rapperswil. This showed only marginal improvements in the percentage of overall correctly predicted cells. Significant improvements were especially achieved in the predictions for cells where there actually are shops. The percentage of exactly predicted cells where at least 1 shop is recorded, nearly doubled. And with a tolerance of +/- 3, the percentage almost reaches an accuracy of 85% for the predictions in Rapperswil.

- Correctly predicted: 97.172% (+ 0.357%)
- Correctly predicted (not 0): 18.571% (+ 8.571%)
- Correctly predicted (not 0, tolerance +/- 1): 65.714% (+ 11.428%)
- Correctly predicted (not 0, tolerance +/- 3): 84.286% (+ 17.143%)

The following plot (figure 1.16) was therefore generated for Rapperswil, which shows how well the model performs on an area which is assumed to be almost completely mapped.

To give a visual indication of how well the prediction performs, the cells are colored according to the following scheme.

Green Difference is less than -1

White Difference is less than or equal to 1

Orange Difference is less than or equal to 3

Red Difference is greater than 3

The color white, which appears transparent in the plot, was added so that the focus lies on the cells where the prediction actually differs from the actual data. Green signifies areas where more shops are actually recorded than predicted, while orange and red mark areas where there are more shops predicted than recorded. The red cells mark the most interesting cells, since in these cells the prediction deviates for more than 3 compared to the actual value. Since all those cells are in the city center of Rapperswil, where the highest density of shops is to be expected and thus larger deviations are more probable, the performance of the model is assumed to be well enough for further predictions.

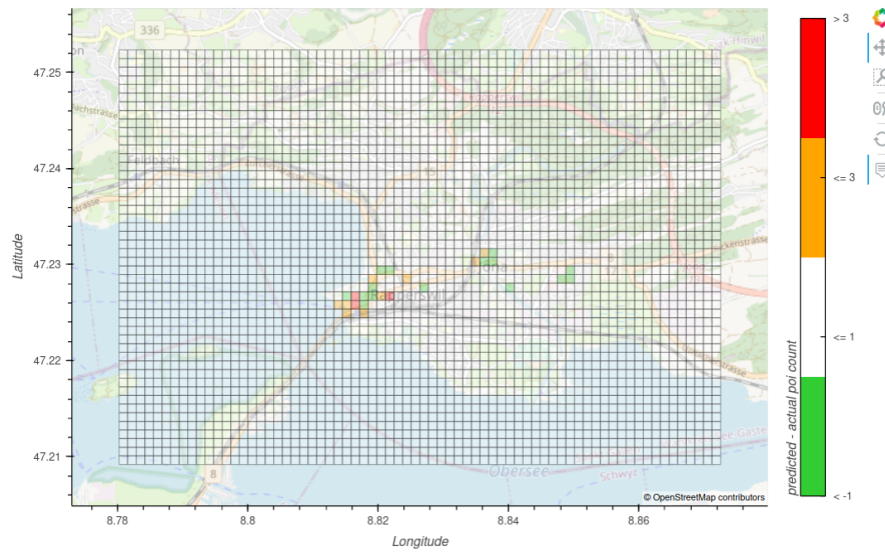


FIGURE 1.16: Prediction for Rapperswil

As a comparison, predictions were made for various selected areas in Switzerland. As examples, figures 1.17 and 1.19 show the results for Bern and Lucerne.

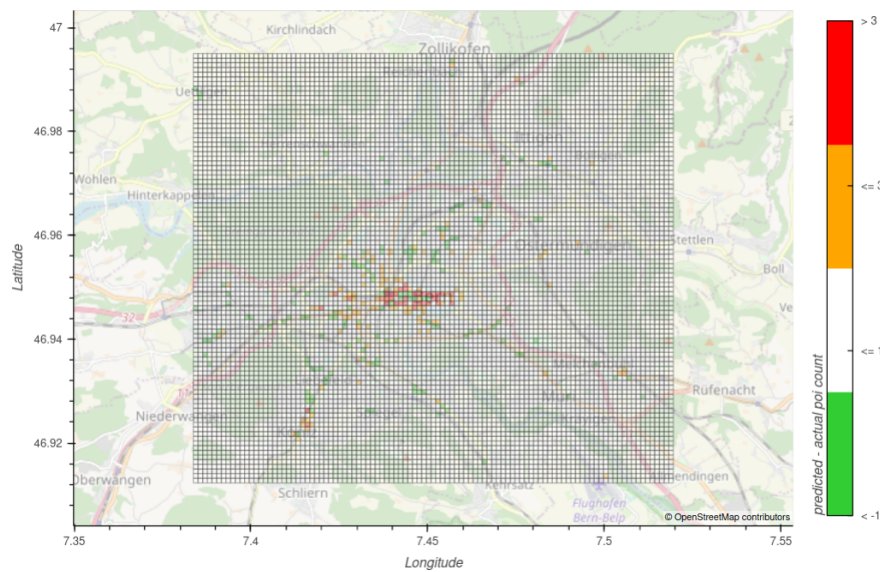


FIGURE 1.17: Prediction of Shops for Bern

For Bern, it is also assumed that shops in this area are almost completely covered. Compared to Rapperswil, however, the proportion of red cells is much higher. This can be explained by the fact that the density of shops in Bern is much greater and the selected tolerance of 3 is quickly exceeded. The comparison of the actual and predicted numbers in figure 1.18 also shows that the prediction predicts a larger concentration of shops at the same locations, but distributes them more evenly across the cells, while in fact the majority of shops are located in a few cells.

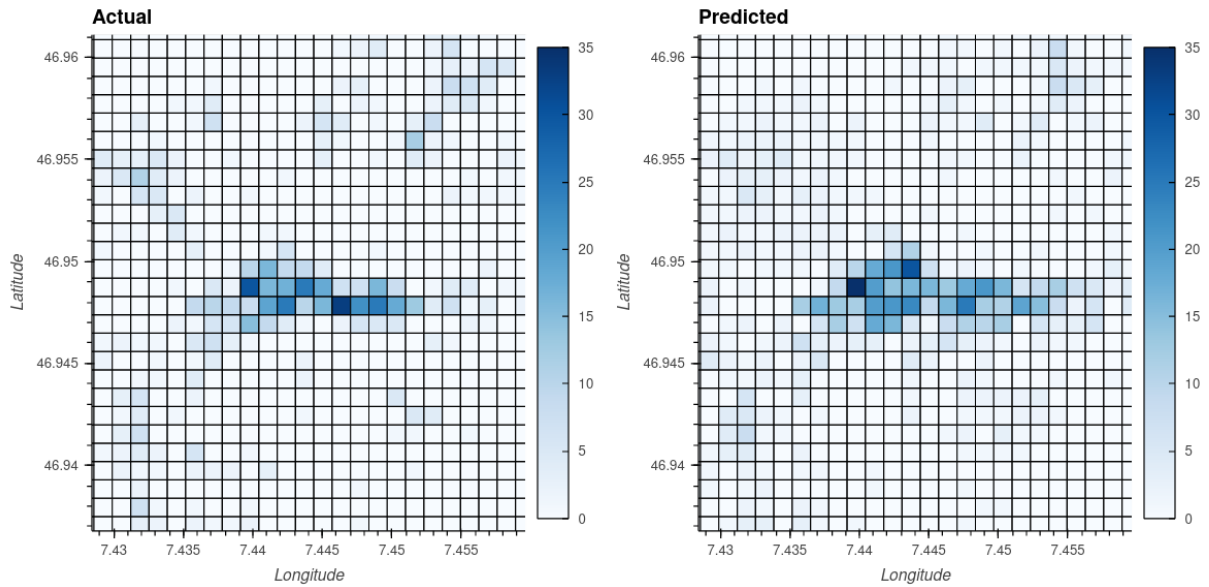


FIGURE 1.18: Comparison Actual and Predicted Shop Count for Bern

As the example of Lucerne clearly shows, the opposite case is also possible. Here you can see that in the center a few red cells are surrounded by a large majority of green cells. This shows that for a large part of the city, fewer shops are expected than are actually recorded.

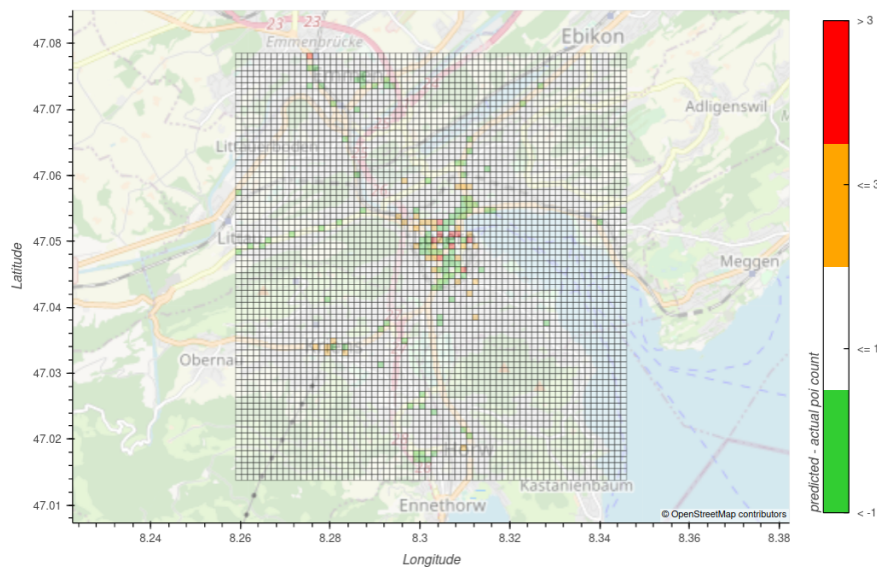


FIGURE 1.19: Prediction for Lucerne

Here, too, the comparison between actual and predicted values provides further information about the distribution (figure 1.20). Thus, shops are predicted at the same locations, but relatively consistently fewer than are actually present. Possible explanations for this are that Lucerne is more completely covered than the training area or that Lucerne, as a tourist stronghold, naturally has a higher density of shops than Zurich.

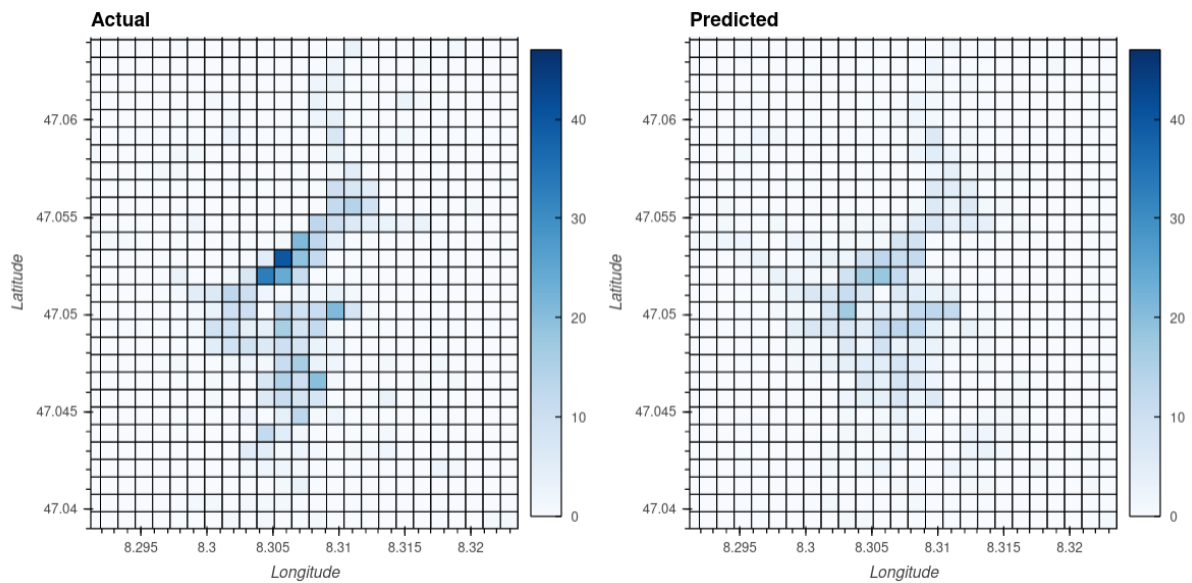


FIGURE 1.20: Prediction for Luzern

Increased Grid Size

A closer look at the results showed that underestimated and overestimated cells often occur in pairs, as shown in figure 1.21 for a section of Rapperswil. One possible reason for this is that the available information is insufficient to make accurate predictions in a 100m grid. This suggested that using a larger grid (in this case 200m) would cause such cells to compensate for each other, resulting in a good value for the prediction.

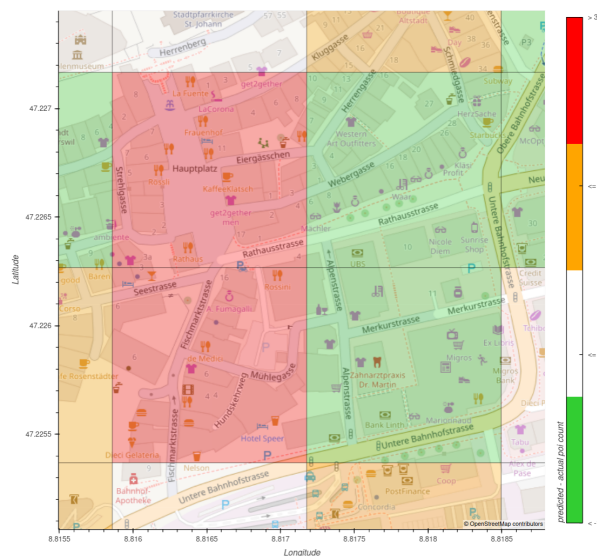


FIGURE 1.21: Prediction for Rapperswil (Zoomed, 100m Grid)

As can be seen from the example for Rapperswil, this is not true in this case. Overestimated and underestimated cells still occur in pairs, with the additional side effect that a small deviation in cells close to each other in a 100m grid is additionally amplified in a 200m grid and thus leads

to an additional inaccuracy. The advantage of the 100m grid is also evident in the generation of the training data. For the same training area, 4x as many data points are available as with the 200m grid, without looking at additional data. Nevertheless, by using a larger grid, the time for data preparation, training, and prediction can be significantly reduced, providing a rough estimate for an area before it is analyzed in detail.

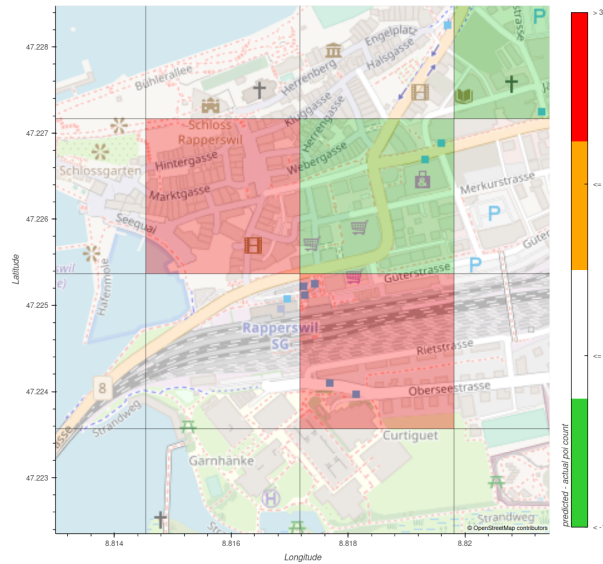


FIGURE 1.22: Prediction for Rapperswil (Zoomed, 200m Grid)

1.5.4 Final Classification

For the final classification of an area regarding its completeness an additional formula was used to calculate a tolerance per cell depending on the number of predicted POIs.

$$d_{tolerance} = \frac{n_{predicted} * 1.3}{10} + 1 \quad (1.2)$$

Formula 1.2 was used to ensure that in cells with a high number of POIs, a larger tolerance is allowed than in areas with only a few POIs. Table 1.4 lists some example values for the tolerances at different prediction values.

TABLE 1.4: Tolerance for Prediction

$n_{predicted}$	$d_{tolerance}$
0 - 3	1
4 - 11	2
12 - 19	3
20 - 26	4
27 - 34	5

Once again, the difference between predicted and actual number of POIs was used as the basis for classification. This difference in relation to the calculated tolerance served as the basis

for the final classification into over-complete, complete, improvable and probably incomplete. For visualization, the cells were colored according to the following classes.

Green Difference is less than $-1 * d_{tolerance}$

White Difference is less than or equal to $d_{tolerance}$

Orange Difference is less than or equal to $2 * d_{tolerance}$

Red Difference is greater than $2 * d_{tolerance}$

Additionally, to give an indication about how the predictions are distributed, each cell is labeled with its predicted number of POIs.

For the validation area Rapperswil, this leads to the representation in figure 1.23. As expected, the model predicts for the vast majority of cells that they are (over-)completely mapped. Only in the center of Rapperswil there are significantly more shops expected than actually recorded.

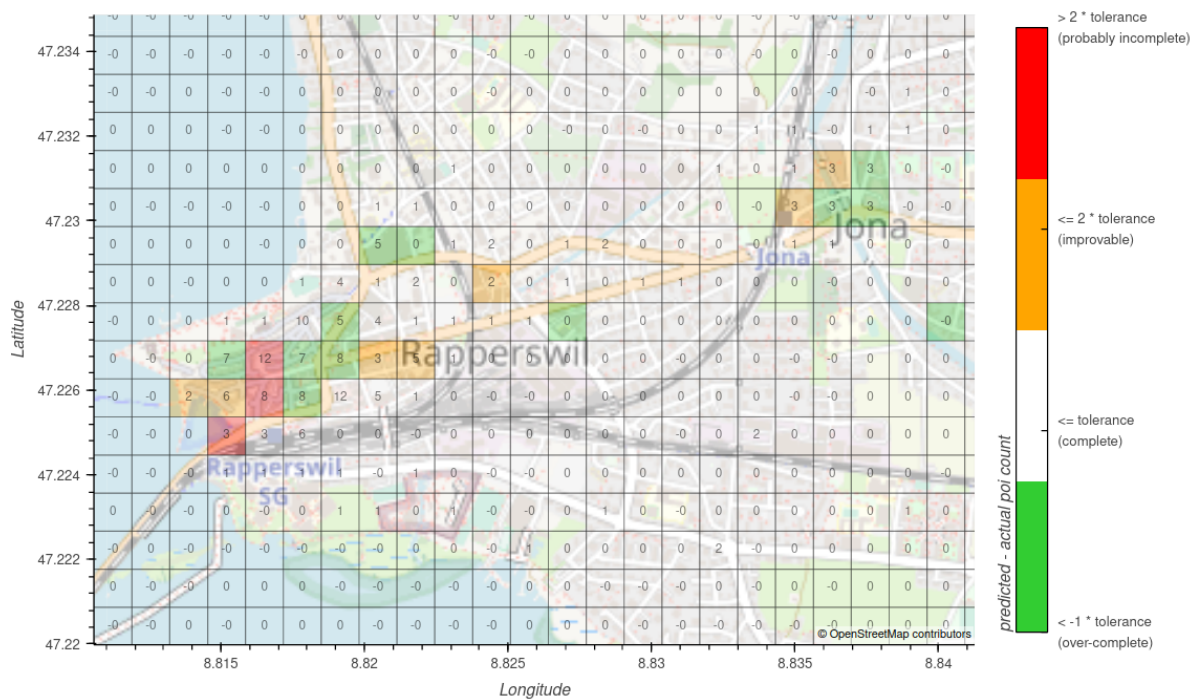


FIGURE 1.23: Estimated Completeness of Shops for Rapperswil

The completeness for our selected prediction areas thus look as follows (table 1.5):

TABLE 1.5: Classification of Shops per Area

Area	Incomplete	Improvable	Complete	Over-Complete
Rapperswil	3 (0.089%)	7 (0.208%)	3335 (99.226%)	15 (0.446%)
Bellinzona	5 (0.018%)	17 (0.063%)	27145 (99.879%)	11 (0.041%)
Bern	33 (0.348%)	80 (0.844%)	9203 (97.119%)	160 (1.689%)
Brugg	2 (0.148%)	6 (0.446%)	1325 (98.586%)	11 (0.819%)
Burgdorf	2 (0.135%)	2 (0.135%)	1471 (99.057%)	10 (0.673%)
Geneva	52 (0.510%)	135 (1.324%)	9941 (97.490%)	69 (0.677%)
Lausanne	36 (0.421%)	62 (0.725%)	8377 (97.908%)	81 (0.947%)
Lucerne	7 (0.147%)	30 (0.631%)	4611 (97.033%)	104 (2.189%)
Solothurn	14 (0.440%)	18 (0.565%)	3127 (98.179%)	26 (0.816%)
Zürich	56 (0.610%)	122 (1.330%)	8726 (95.096%)	272 (2.964%)

A full set of plots for the areas mentioned in table 1.5 can be found in appendices A and B.

1.5.5 Limitations

As expected, the results depend heavily on the quality of the data already available. Statements about the completeness of a selected region are therefore dependent on the completeness of the training area. However, since only intrinsic data are used and no sufficiently large area is known to be completely covered, predictions can only be based on data that are assumed to be rather complete.

Furthermore, the model can only be applied to areas for which data has already been collected. In the hypothetical case of a city for which no data has been collected yet, no or only few additional POIs could be predicted, since the intrinsic correlations cannot be drawn. This problem is already apparent for the selected prediction areas in the more rural regions. As expected, the density of POIs is lower there and therefore a precise prediction is correspondingly more difficult.

The whole prediction depends on the assumption that the training area is well - or ideally - completely mapped. If there was a large enough area for which it is known that it is completely mapped, that data could serve as a 'golden data set' which presumably would enable more precise predictions.

1.5.6 Target Achievement

In a first step, it could be shown that an approximate estimate of completeness is already possible by means of aerial photography and the total number of POIs recorded. However, a further pursuit of this approach is not recommended, since the results become worse the more granular the POIs are divided into categories. In addition, the computational effort for image data is many times greater and therefore cannot be easily repeated on home devices. Since aerial photographs with the resolution used for this project are only available for Switzerland, the model can also only be used to predict areas in Switzerland.

Additionally, for several selected areas it could be shown that a prediction of the completeness of the recorded POIs can be approximated with intrinsic data. For areas where it is assumed that they are more likely to be completely covered, the model provides a prediction that is mostly consistent with the expected values. This model can be applied to areas within similar geographical environments. Further, we provided the means to enrich the training data with additional intrinsic and extrinsic data to improve the results as can be seen with the use of primary land usage and land coverage which was also calculated from intrinsic data and population data which was provided by a third party.

The range of predictable areas could easily be changed by training with data from another geographical region, which would allow for predictions in the respective area. The data pipeline is easily adaptable and could thus also be modified to use data from an external data source as training data for example if somewhere a 'golden data set' for a certain region was provided.

1.5.7 Demo

Since the demo only serves as a proof of concept and an easily usable application without the need for time-consuming calculations, the demo also has potential for improvements. The UX is minimal, optically not that appealing, and not responsive. No visual improvements are proposed other than using a metric coordinate reference system for more straightforward interpretation of the plots. The problem that it only works with prepared data is not trivial to overcome. One could allow small areas to be chosen and predict them. If the area is too small, the prediction is not as good, and as the area gets larger, the computation time grows. For this outlook, two possibilities were discussed. A large part of the computation is not the prediction but data preparation. So it would be possible to prepare data for a large area and filter it down for the user's chosen area. For an area like Switzerland, this should be doable but could not be verified as the used servers were not powerful enough (4 core at 3.4GHz, 16GB RAM) and resulted in Kernel Panics.

However, this is certainly not a solution that scales for larger areas. Also, it must be updated and recalculated periodically to remain valid. Nevertheless, it could serve multiple users without a powerful server as only the periodical data preparation is computationally intensive. The other possibility is to optimize the data preparation as proposed in a previous section. With enough optimization, it would be possible to calculate areas of reasonable size. The tradeoff would be the higher base computational power needed.

1.6 Outlook, Further Development

The project offers some opportunities for further development, which were not pursued due to time constraints, but could further improve the quality of the predictions.

1.6.1 Data Enhancements

In highly dense areas, the three-dimensionality could also be a deciding factor for the number of POIs. Thus, the level of buildings or the usable building area could be used. Both values are available in OSM, although the number of building levels is only captured in certain regions.

Similar to the calculation of the LULC data, a value for the usability of a cell could be defined. For example, cells located on a steep slope could receive a low usability value due to the steep slope, while a flat area near a body of water, for example, would receive a higher value. It is suspected that this could improve the prediction also for non-urban areas, as is the case for a large part of Switzerland.

Additional external data

1.6.2 Performance Optimizations

Once running reasonably fast for this project's training area, the data preparation was not further optimized. Reasonable in this context means less than 30 minutes. For example, one could use Spatialpandas instead of GeoPandas. Spatialpandas can work with Dask data frames. This enables the parallelization of buffer and nearest neighbor calculations. If calculations on a multicore processor is not fast enough one could even use Dask in a cluster. Generally, the project offers many opportunities for parallelization since almost all calculations done with geographical data could be divided into smaller chunks.

Bibliography

- Barron, C., Neis, P., & Zipf, A. (2014). A comprehensive framework for intrinsic openstreetmap quality analysis. *Transactions in GIS*, 18(6), 877–895. <https://onlinelibrary.wiley.com/doi/abs/10.1111/tgis.12073>
- Fritz, O., Auer, M., & Zipf, A. (2021). Entwicklung eines regressionsmodells für die vollständigkeitsanalyse des globalen openstreetmap-datenbestands an nahverkehrs-busstrecken. *AGIT-Journal für Angewandte Geoinformatik*.
- Hecht, R., Kunze, C., & Hahmann, S. (2013). Measuring completeness of building footprints in openstreetmap over space and time. *ISPRS International Journal of Geo-Information*, 2(4), 1066–1091.
- Howard, J., & Gugger, S. (2020). *Deep learning for coders with fastai and pytorch*. O'Reilly Media.
- Kaur, J., & Singh, J. (2018). An automated approach for quality assessment of openstreetmap data. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 707–712. <https://doi.org/10.1109/GUCON.2018.8674899>
- Kaur, J., Singh, J., Sehra, S. S., & Rai, H. S. (2017). Systematic literature review of data quality within openstreetmap. *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, 177–182. <https://doi.org/10.1109/ICNGCIS.2017.35>
- Openstreetmap wiki*. (2021). Retrieved March 7, 2021, from <https://wiki.openstreetmap.org/wiki/Completeness>
- Tobin, J., Karayev, S., Abbeel, P., Sher, S., & Le, J. (2019). *Full stack deep learning*. Retrieved March 7, 2021, from <https://fall2019.fullstackdeeplearning.com>

Glossary

base model Simple model that serves as a baseline to which complex solutions are compared in order to verify better results.

docker Open-source software to isolate applications using container virtualization.

fast.ai Open-source library for deep learning.

feature Abstraction of a real world phenomenon.

golden data set A set of data containing clean, validated and complete data.

intrinsic basic to a thing; in the context of this thesis: data from within OSM.

Jupyter Notebook Web-based interactive environment allowing the creation of documents containing text, live code, and visualizations.

multipolygon Relations of type multipolygon are used to represent complex areas with holes inside or consisting of multiple disjoint areas.

polygon Relations of type polygon are used to represent simple areas.

Resnet34 A convolutional neural network with 34 layers the can be used as image classification model. The model has been pretrained on the ImageNet dataset.

tag A tag consists of two items, a key and a value. Tags describe specific features of map elements (nodes, ways, or relations) or changesets.

XGBoost Open-source library providing a regularizing gradient boosting framework.

Acronyms

LULC Land Usage and Land Coverage. 3, 32

MAE Mean Absolute Error. 21

MSE Mean Squared Error. 21

MSLE Mean Squared Logarithmic Error. 21

OSM OpenStreetMap. 1, 3–5, 7, 9, 31

PBF Protocolbuffer Binary Format. 9, 13, 14

POI Point of Interest. 1–3, 6, 12, 18, 28–31

RMSE Root Mean Squared Error. 21

Appendix A

Completeness of Shops for Selected Areas

The plots in this appendix show the plots for the resulting completeness estimations as described in subsection 1.5.4 and serve as additional examples. Additional information about the calculation of the tolerance and the interpretation of the plots is depicted in subsection 1.5.4 and will thus not be repeated for each plot.

Rapperswil

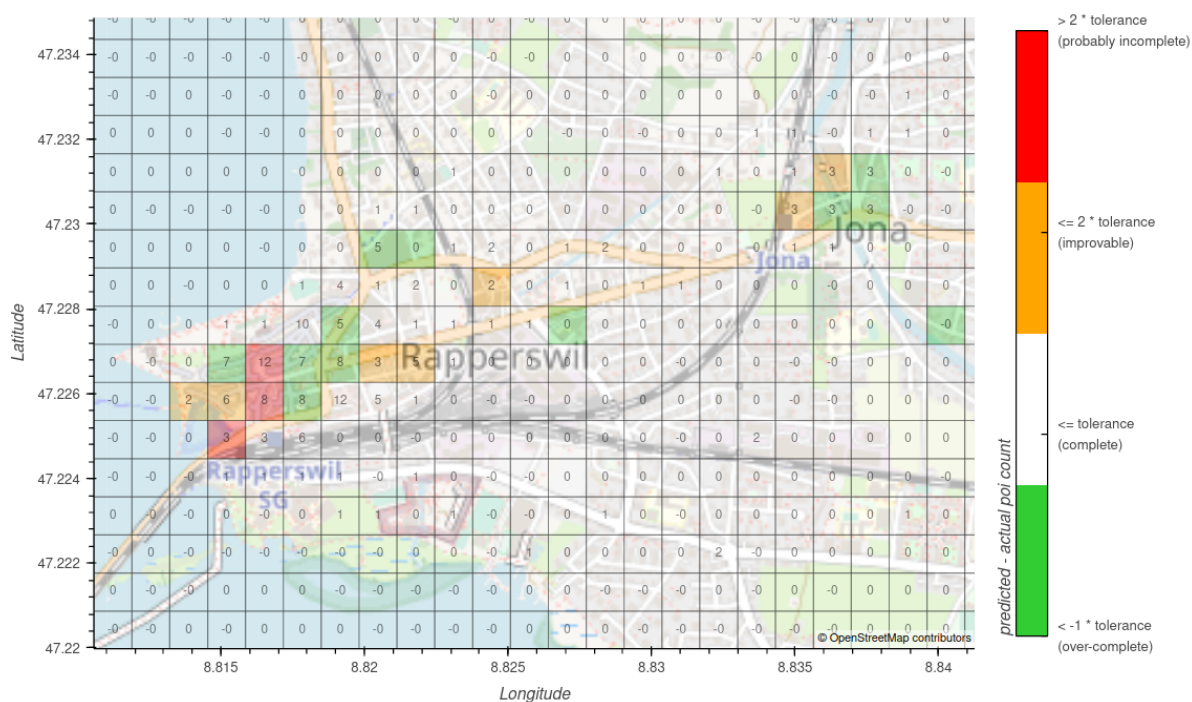


FIGURE A. 1: Estimated Completeness of Shops for Rapperswil

Bellinzona

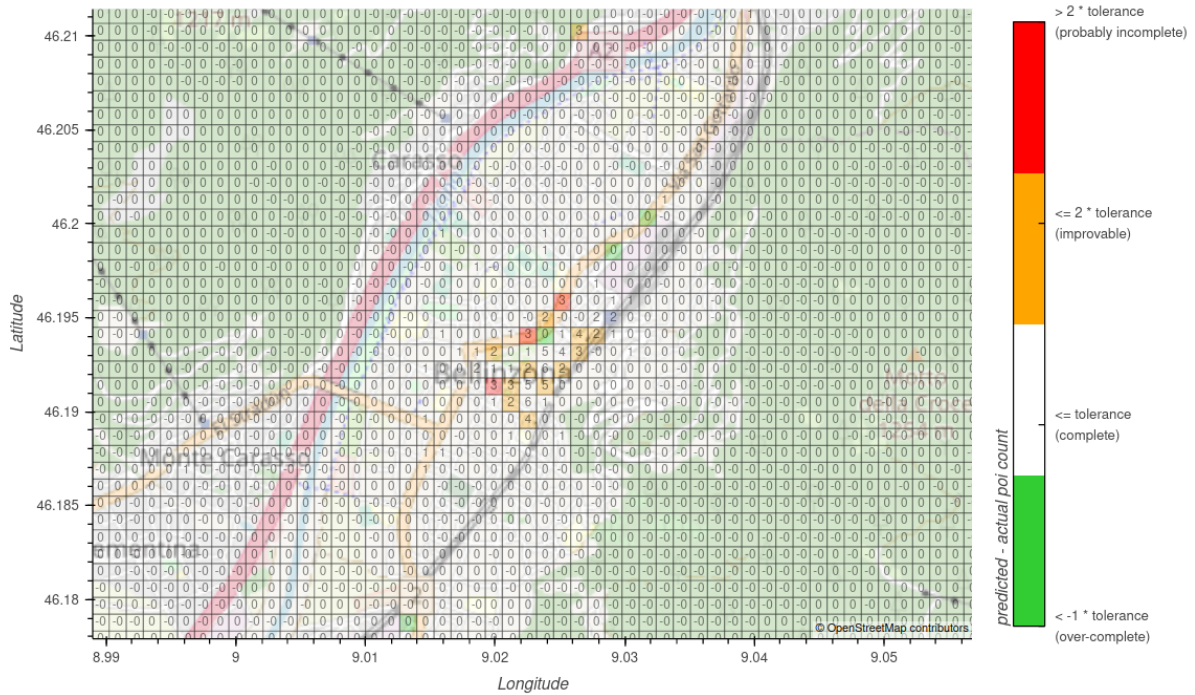


FIGURE A.2: Estimated Completeness of Shops for Bellinzona

Bern

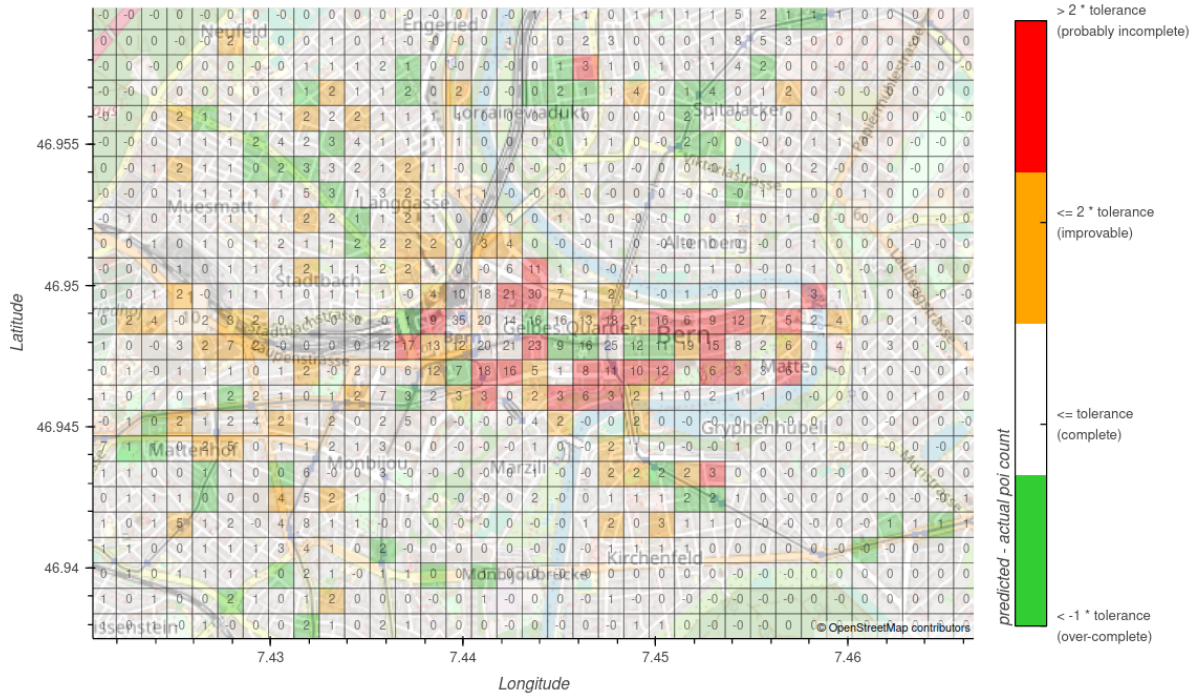


FIGURE A.3: Estimated Completeness of Shops for Bern

Brugg

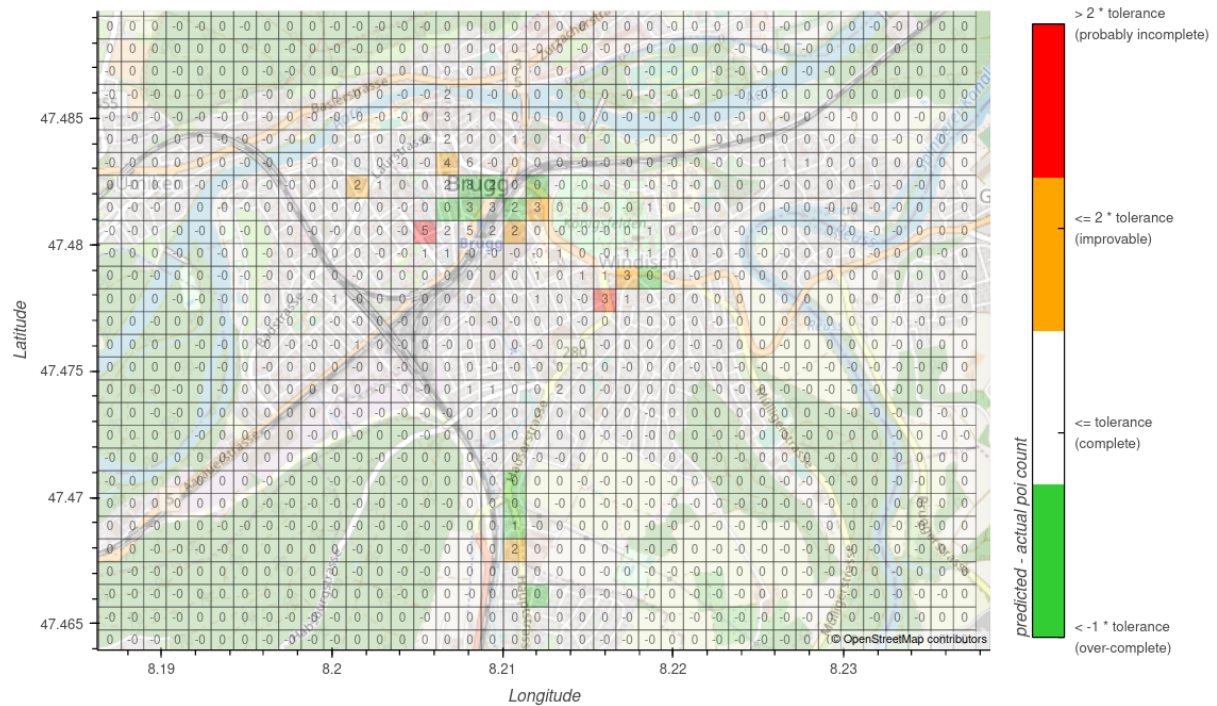


FIGURE A.4: Estimated Completeness of Shops for Brugg

Burgdorf

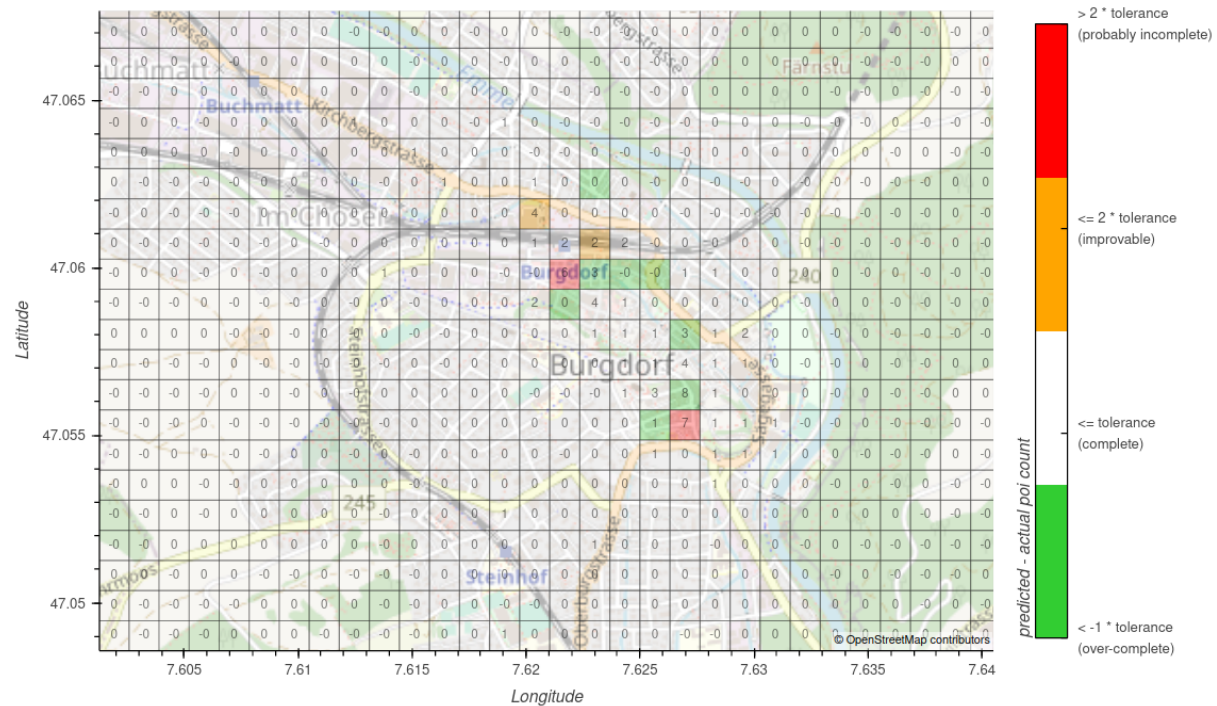


FIGURE A.5: Estimated Completeness of Shops for Burgdorf

Geneva

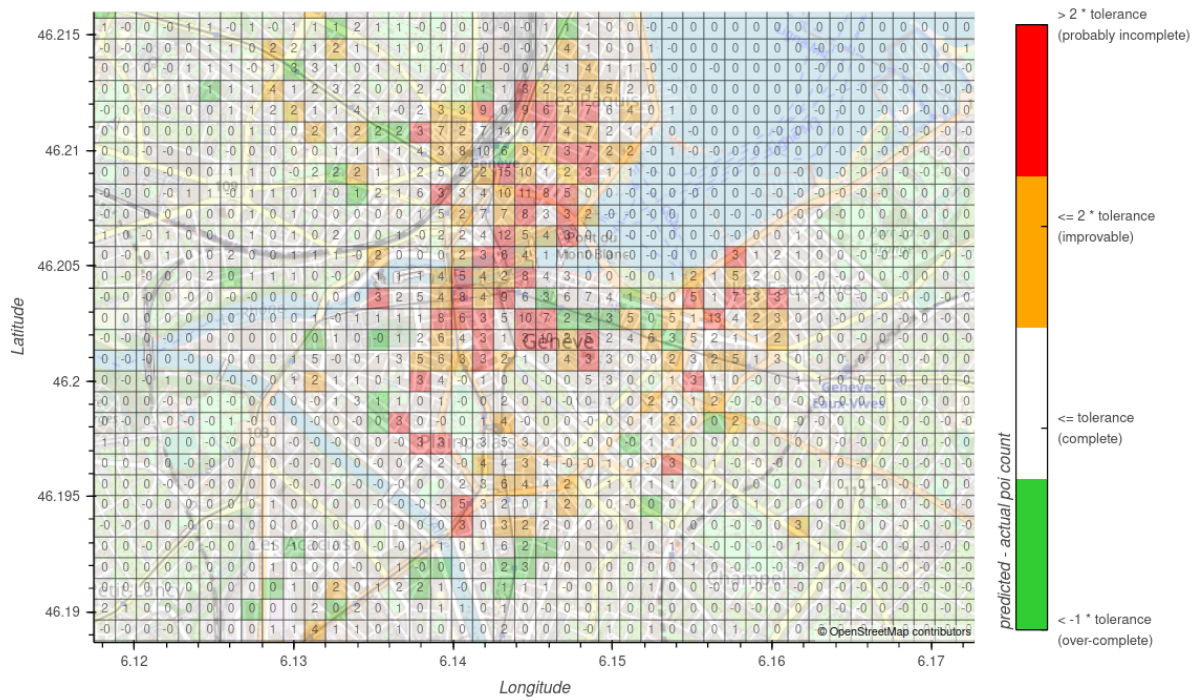


FIGURE A.6: Estimated Completeness of Shops for Geneva

Lausanne

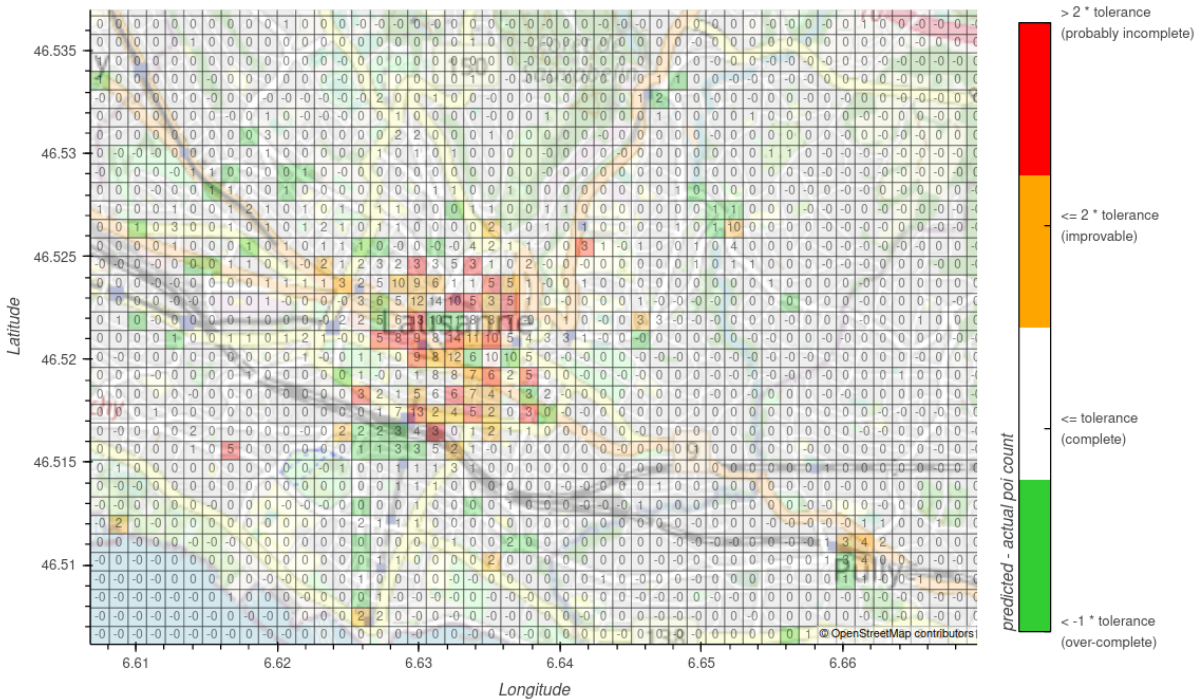


FIGURE A.7: Estimated Completeness of Shops for Lausanne

Lucerne

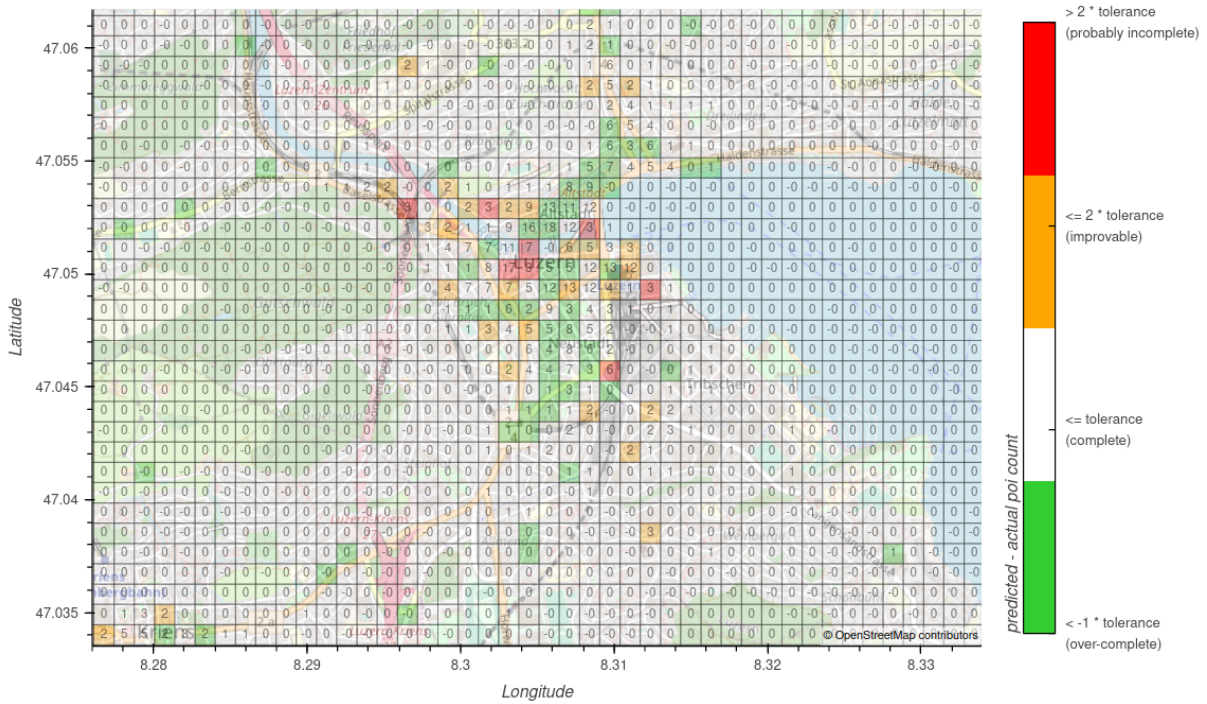


FIGURE A.8: Estimated Completeness of Shops for Lucerne

Solothurn

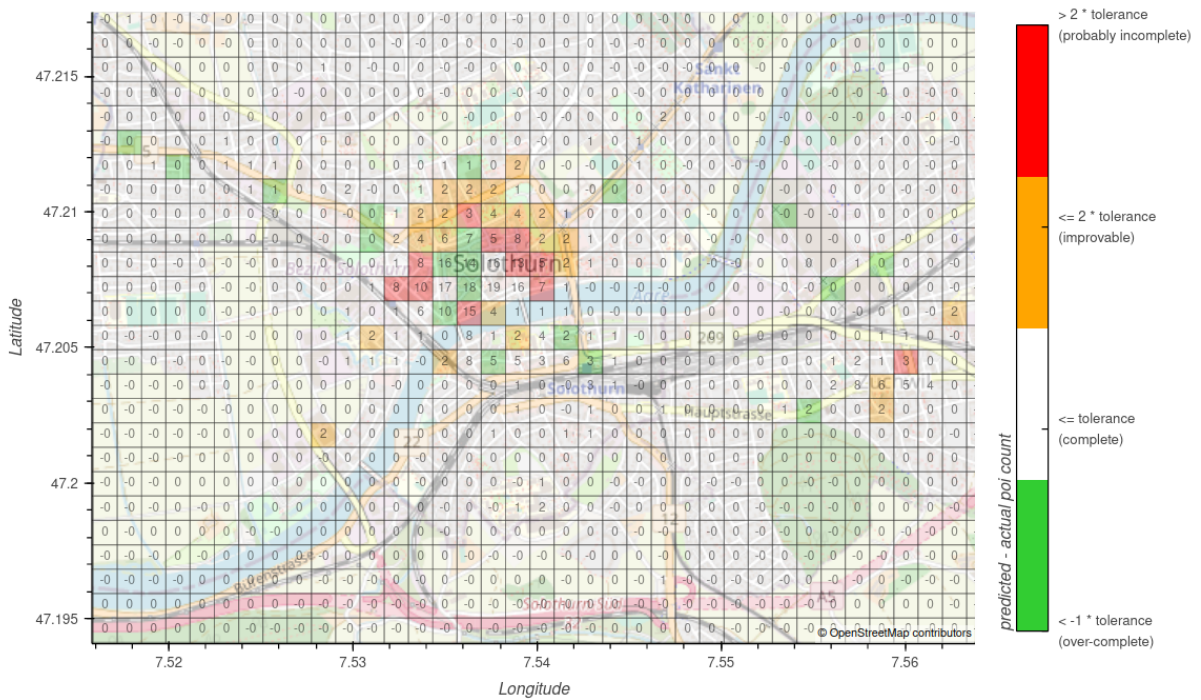


FIGURE A.9: Estimated Completeness of Shops for Solothurn

Zürich

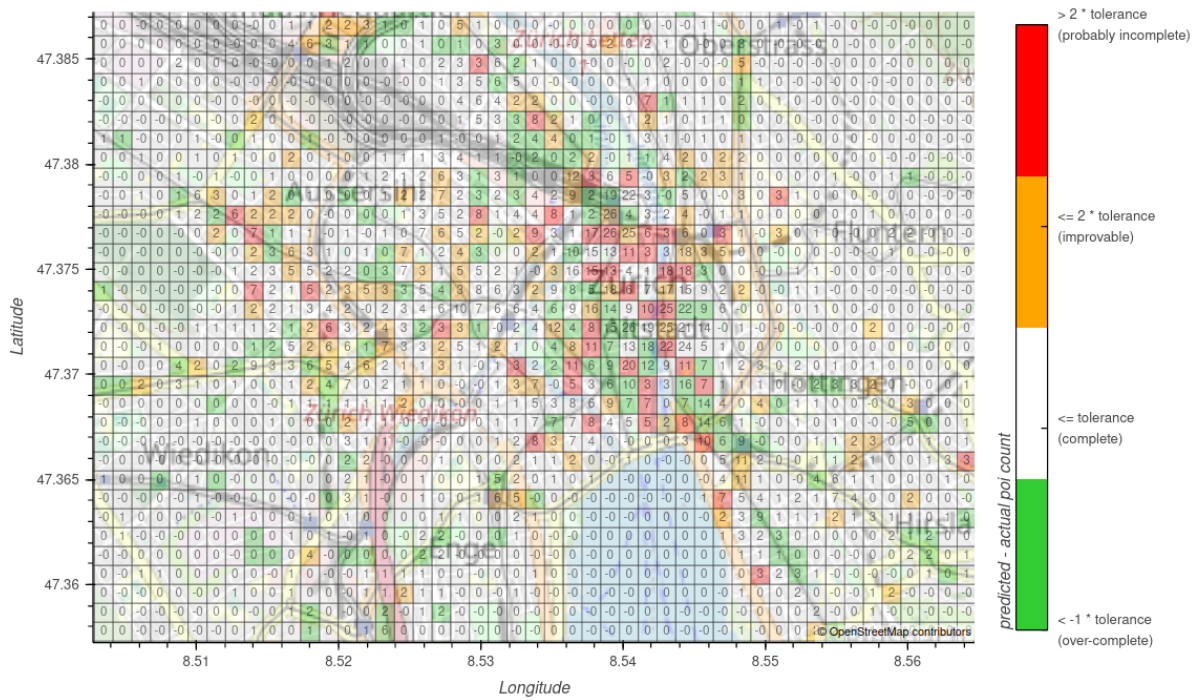


FIGURE A.10: Estimated Completeness of Shops for Zürich

Appendix B

Comparison of Actual and Predicted Shop Counts for Selected Areas

The plots in this appendix show the difference between the actual shop counts (left) and the shop count predicted by the proposed model (right). The plots serve as additions to the plots presented in 1.5.3.

Rapperswil

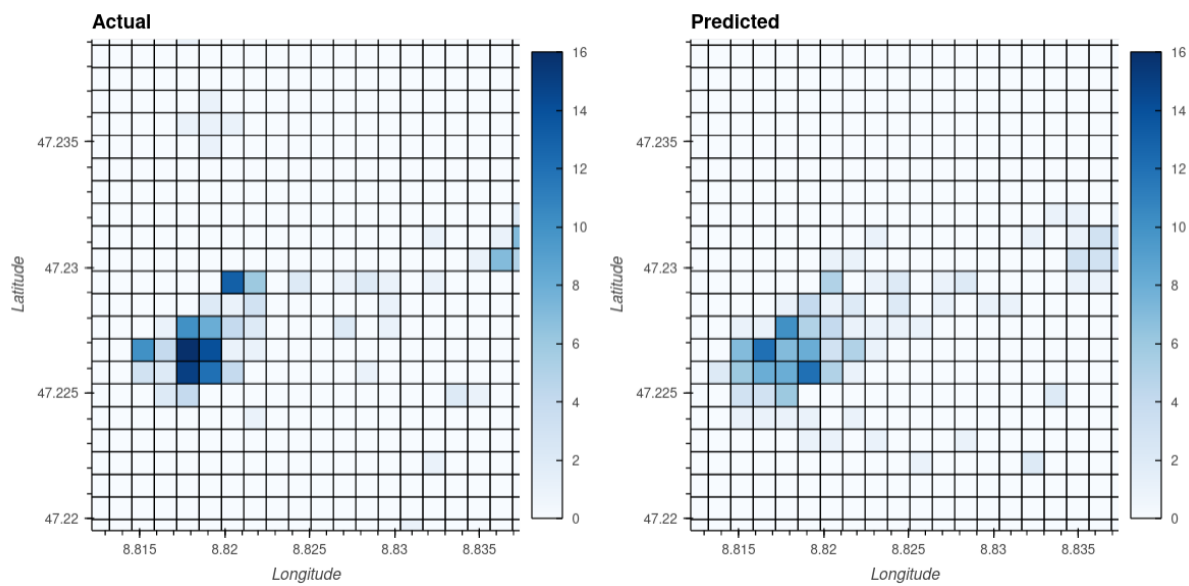


FIGURE B.1: Comparison Actual and Predicted Shop Count for Rapperswil

Bellinzona

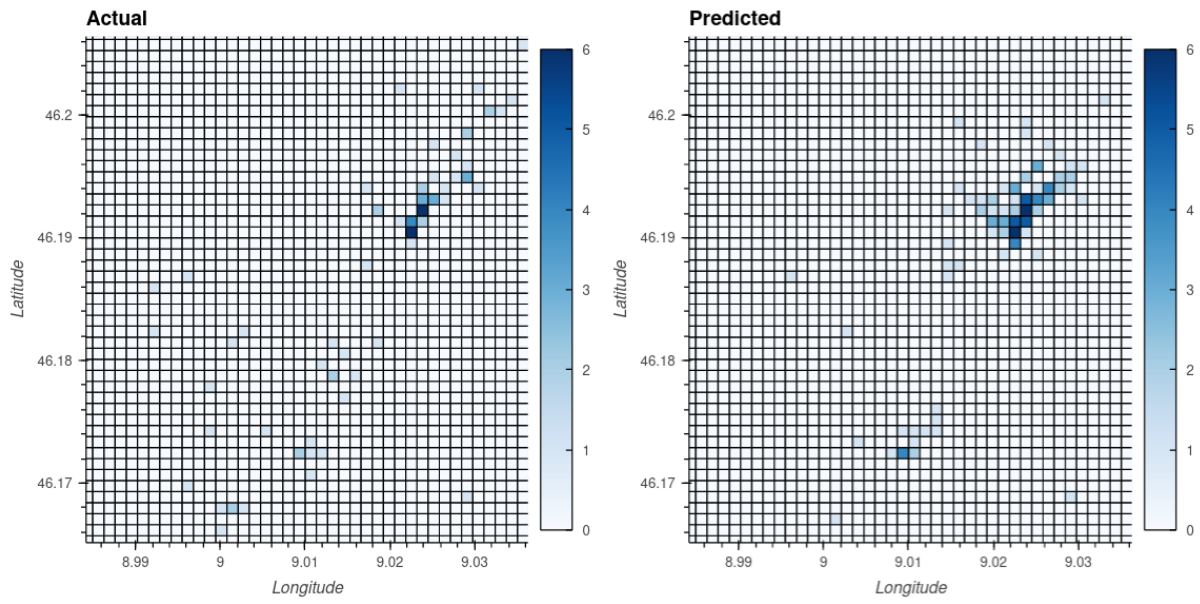


FIGURE B.2: Comparison Actual and Predicted Shop Count for Bellinzona

Bern

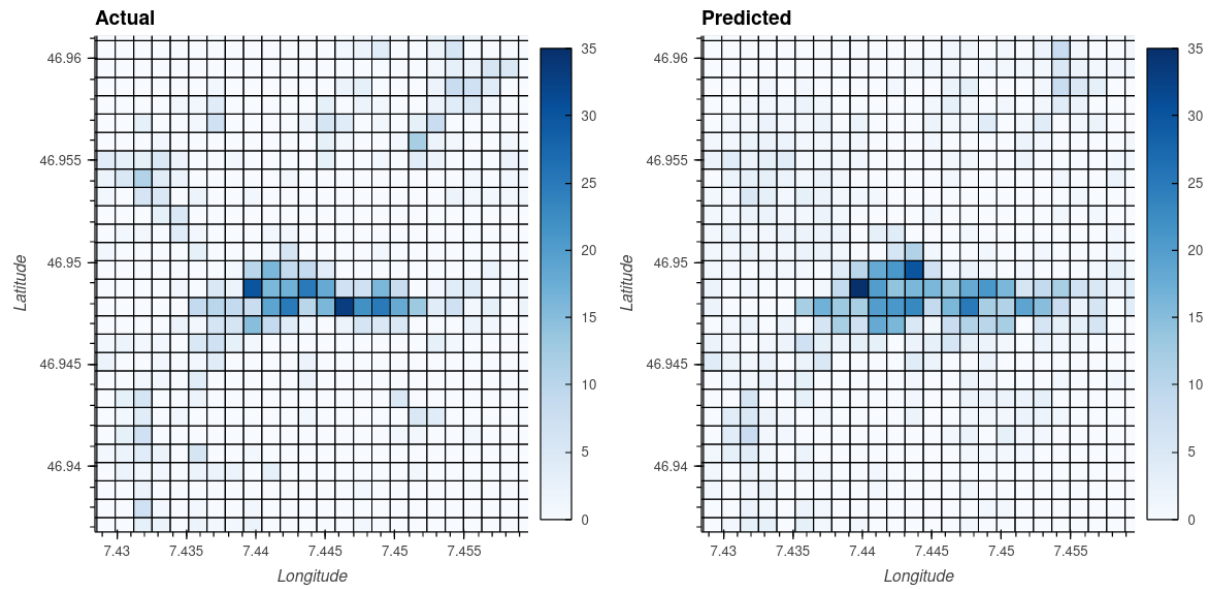


FIGURE B.3: Comparison Actual and Predicted Shop Count for Bern

Brugg

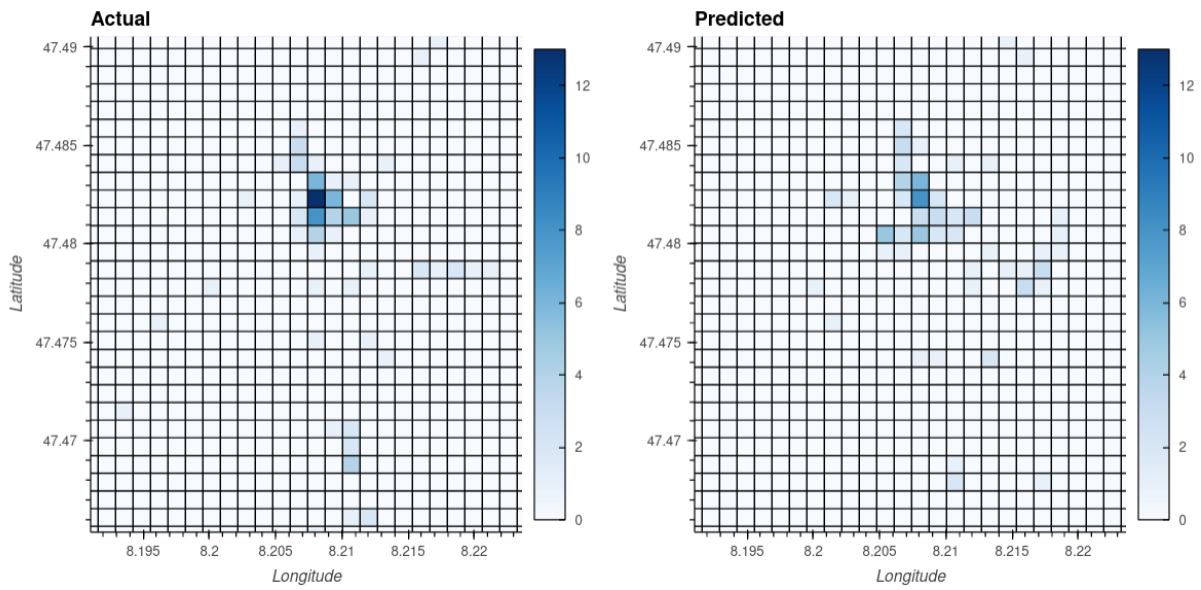


FIGURE B.4: Comparison Actual and Predicted Shop Count for Brugg

Burgdorf

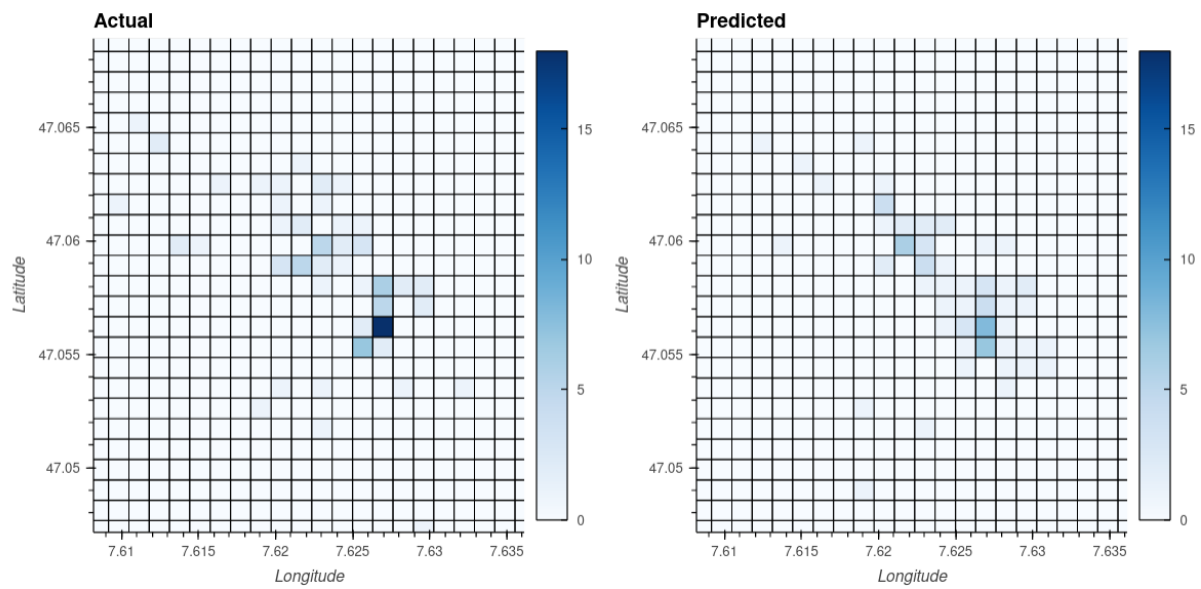


FIGURE B.5: Comparison Actual and Predicted Shop Count for Burgdorf

Geneva

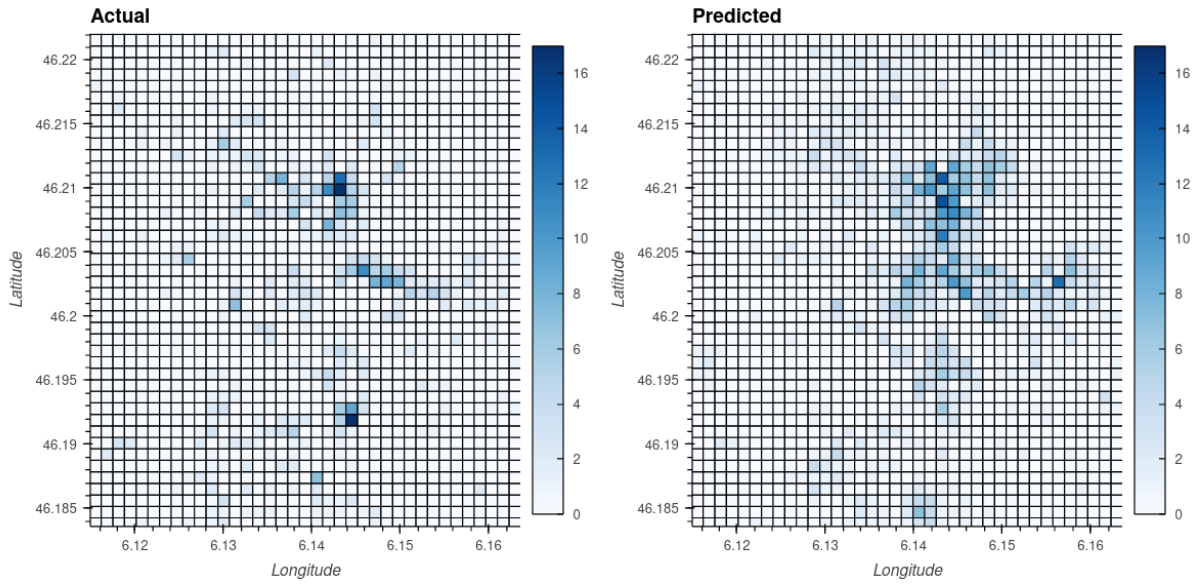


FIGURE B.6: Comparison Actual and Predicted Shop Count for Geneva

Lausanne

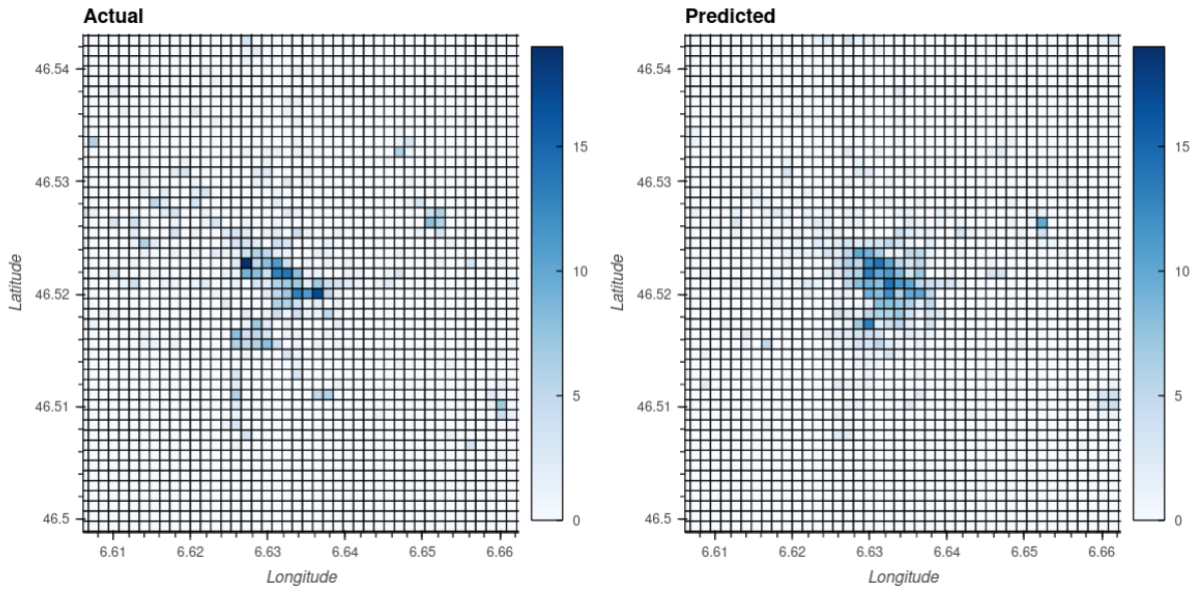


FIGURE B.7: Comparison Actual and Predicted Shop Count for Lausanne

Lucerne

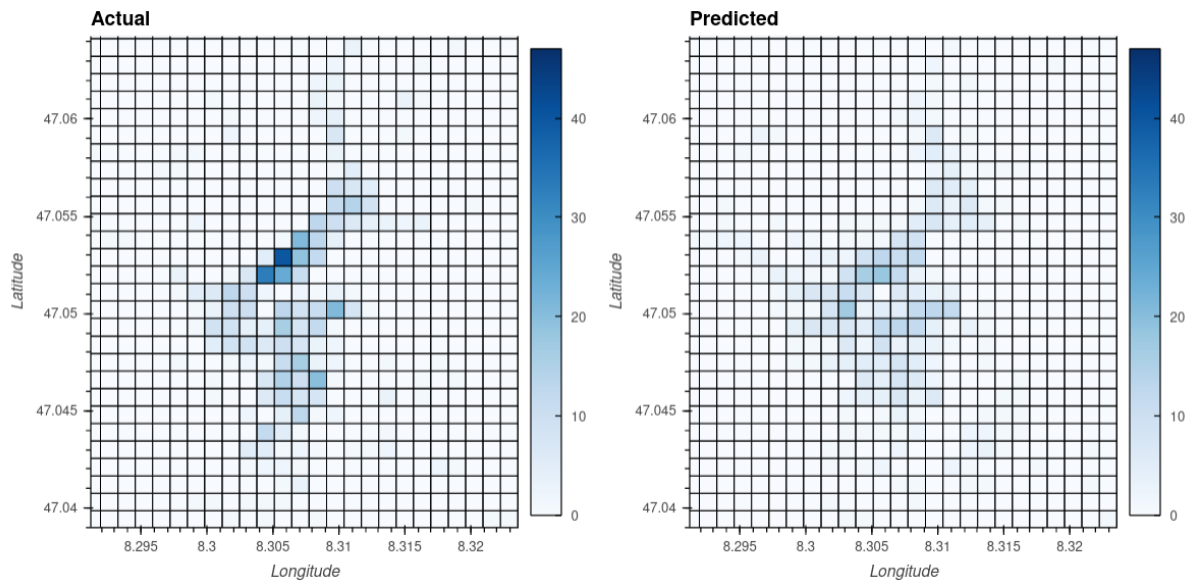


FIGURE B.8: Comparison Actual and Predicted Shop Count for Lucerne

Solothurn

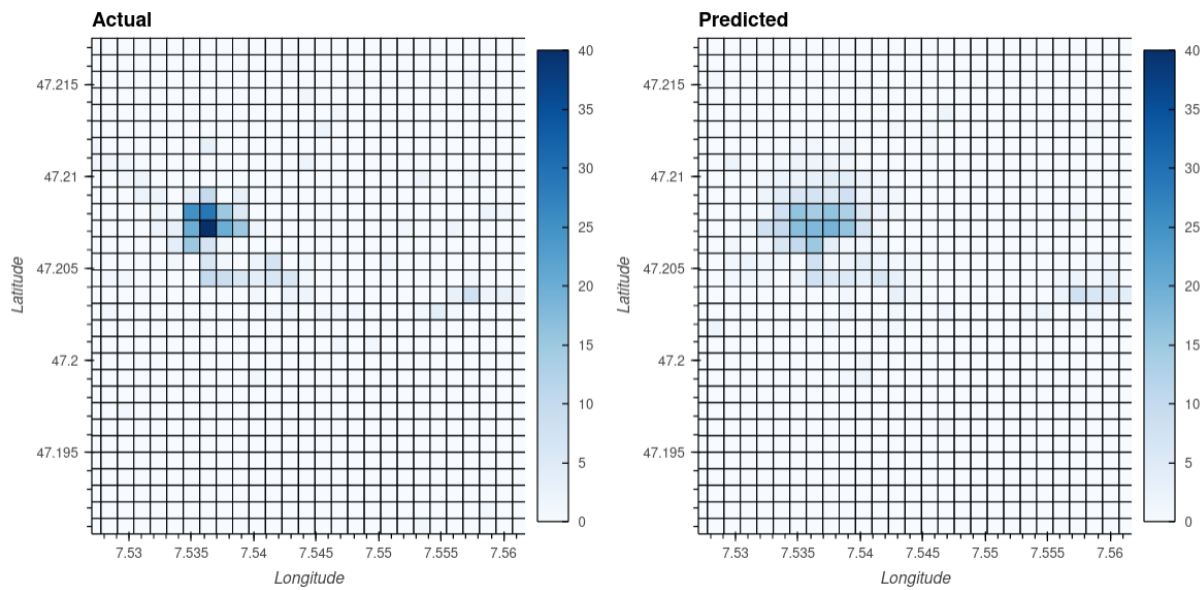


FIGURE B.9: Comparison Actual and Predicted Shop Count for Solothurn

Zürich

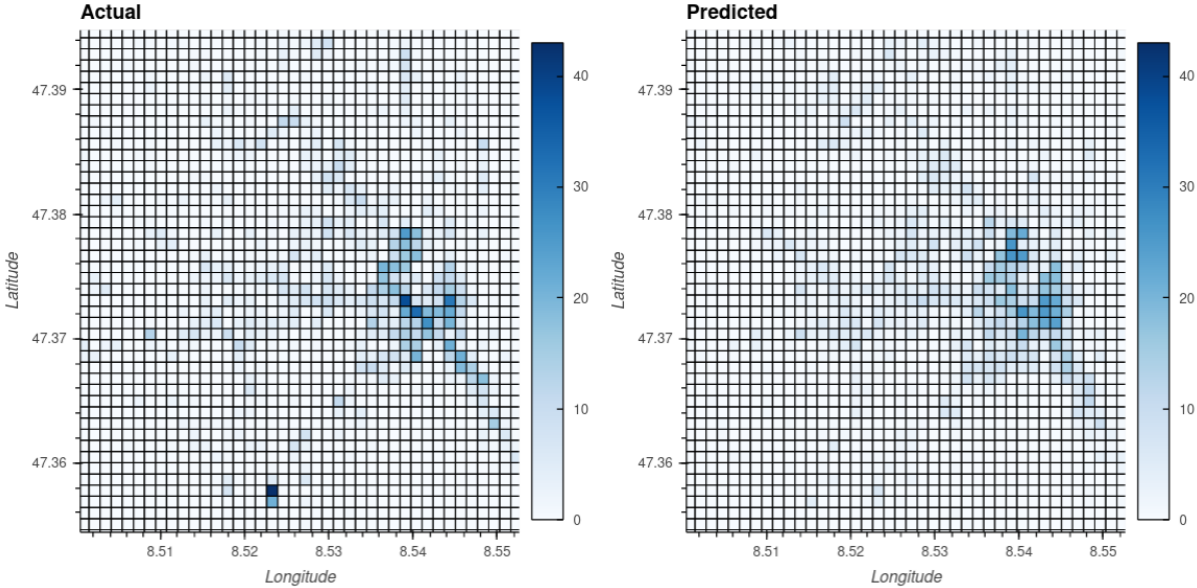


FIGURE B.10: Comparison Actual and Predicted Shop Count for Zürich

Appendix C

Software Documentation

Installation

For higher portability and easier dependency management, docker containers with docker-compose are used. This limits the prerequisites for installing the project on a new machine to docker and docker-compose. To train the models efficiently, GPU acceleration is recommended, and the docker images are designed to use it. For Nvidia GPUs, the Nvidia driver, Nvidia Cuda driver, and the Nvidia Container Toolkit are needed on the host machine. AMD GPUs were not tested but should at least in theory work in a similar manner. It must be mentioned that ROCm, AMDs alternative to CUDA, is at the time of writing still in beta. The docker-compose file consists of four services and is basically the nbdev template modified to match the requirements of this project.

- notebook: The jupyter notebook
- watcher: Watches the notebooks for change and triggers the documentation rebuild
- jekyll: A webserver to host a local version of the documentation
- demo: The panel server that hosts the demo

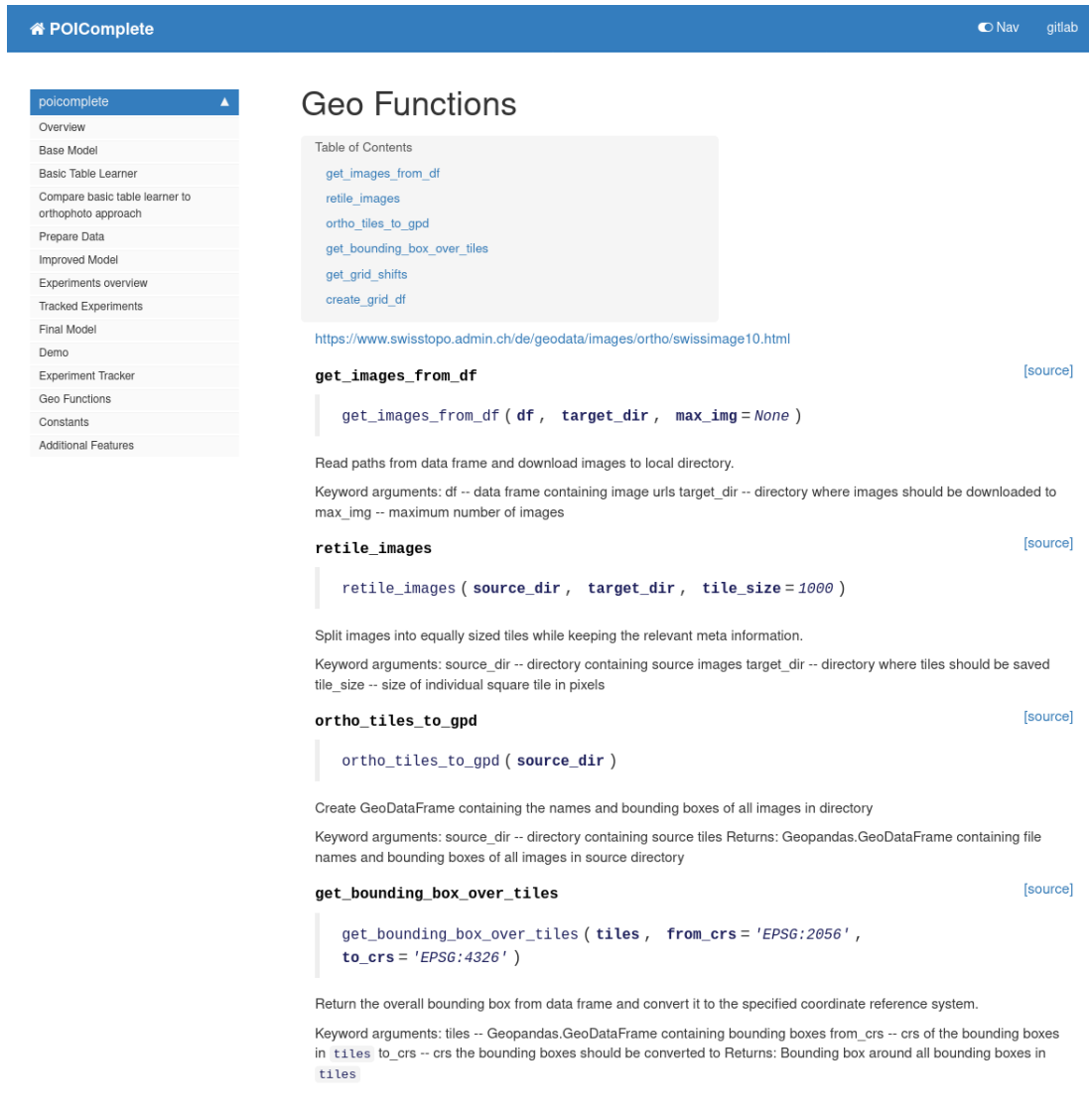
Tutorial / User Manual

A separate user manual was not written. Nevertheless, thanks to the use of notebooks, the code, and its usage are documented in them.

Also, is there no dedicated tutorial for this project. The notebooks are numbered. To explore the project in the way it was built, one should start at one and work through to ten. To only check the results or train on own data, the notebook `10_final_model` is enough. The notebook `00_core` was the first notebook to explore the project. It is there for the sake of completeness but has no further importance. The notebooks `70_demo` is the one that gets served by panel for the demo. The notebook `80_time_tracking` is used to generate the charts for the time tracking in the chapter software documentation. The notebooks starting with the 90s are utility notebooks that get exported as libraries and reused in the other notebooks. This is done with nbdev.

Reference Manual

With nbdev, it is also possible to generate code documentation. It automatically exports classes, functions, cells, and output to a static website. The code can be searched and is linked to the repository. In addition, it parses python docstrings and the function signature. Therefore, every argument is well documented, including default values. This was used instead of pydoc as it is better suited for notebooks.



The screenshot shows a web interface for 'POIComplete' with a navigation menu on the left and a main content area titled 'Geo Functions'. The navigation menu includes: Overview, Base Model, Basic Table Learner, Compare basic table learner to orthophoto approach, Prepare Data, Improved Model, Experiments overview, Tracked Experiments, Final Model, Demo, Experiment Tracker, Geo Functions (selected), Constants, and Additional Features. The main content area features a 'Table of Contents' with links to: `get_images_from_df`, `retiler_images`, `ortho_tiles_to_gpd`, `get_bounding_box_over_tiles`, `get_grid_shifts`, and `create_grid_df`. Below this is a URL: <https://www.swisstopo.admin.ch/de/geodata/images/ortho/swissimage10.html>. The first function, `get_images_from_df`, is detailed with its signature: `get_images_from_df (df , target_dir , max_img = None)`. Its description reads: 'Read paths from data frame and download images to local directory.' Keyword arguments are: `df` -- data frame containing image urls, `target_dir` -- directory where images should be downloaded to, `max_img` -- maximum number of images. A '[source]' link is provided. The second function, `retiler_images`, has signature: `retiler_images (source_dir , target_dir , tile_size = 1000)`. Description: 'Split images into equally sized tiles while keeping the relevant meta information.' Keyword arguments: `source_dir` -- directory containing source images, `target_dir` -- directory where tiles should be saved, `tile_size` -- size of individual square tile in pixels. A '[source]' link is provided. The third function, `ortho_tiles_to_gpd`, has signature: `ortho_tiles_to_gpd (source_dir)`. Description: 'Create GeoDataFrame containing the names and bounding boxes of all images in directory.' Keyword arguments: `source_dir` -- directory containing source tiles. Returns: `Geopandas.GeoDataFrame` containing file names and bounding boxes of all images in source directory. A '[source]' link is provided. The fourth function, `get_bounding_box_over_tiles`, has signature: `get_bounding_box_over_tiles (tiles , from_crs = 'EPSG:2056' , to_crs = 'EPSG:4326')`. Description: 'Return the overall bounding box from data frame and convert it to the specified coordinate reference system.' Keyword arguments: `tiles` -- `Geopandas.GeoDataFrame` containing bounding boxes, `from_crs` -- crs of the bounding boxes in `tiles`, `to_crs` -- crs the bounding boxes should be converted to. Returns: Bounding box around all bounding boxes in `tiles`. A '[source]' link is provided.

FIGURE C.1: nbdev generated code documentation

Appendix D

Project Management

TABLE D.1: Links and Tools

Name	URL
Project Repository	https://gitlab.com/mcrisafu/mapcomplete
Project Documentation	https://gitlab.com/mcrisafu/mapcomplete_doc
Task	https://wiki.ost.ch/x/hAhjAw
Preliminary Project	https://gitlab.com/geometalab/osmcross

Prototypes, Releases, Milestones

Due the experimental nature of the project the classical approach of one or two prototypes, an MVP and fixed releases was not ideal. Therefore, only milestones as shown in table D.2 were defined. They are mainly based on fixed deadlines set by the school.

TABLE D.2: Milestones and Deadlines

Nr.	Date	Description
M	25.02.2021	Kickoff-Meeting
M	24.03.2021	Initial Project Planning
M	25.03.2021	Preparation and statistical analysis of the data
M	31.03.2021	Implementation Base Model
M	22.04.2021	Intermediate Presentation
M	21.05.2021	Feature Freeze
M	04.06.2021	Project Finished (Must-Have)
D	09.06.2021	Abstract Submission
D	15.06.2021	Abstract Release
D	16.06.2021	Print Report
M	18.06.2021	End of Project

Team, Roles and Responsibilities

The project is implemented by Marco Crisafulli and Dominic Monzón and supervised by Prof. Stefan F. Keller and Nicola Jordan. Figure D.1 shows the organizational structure of the project.

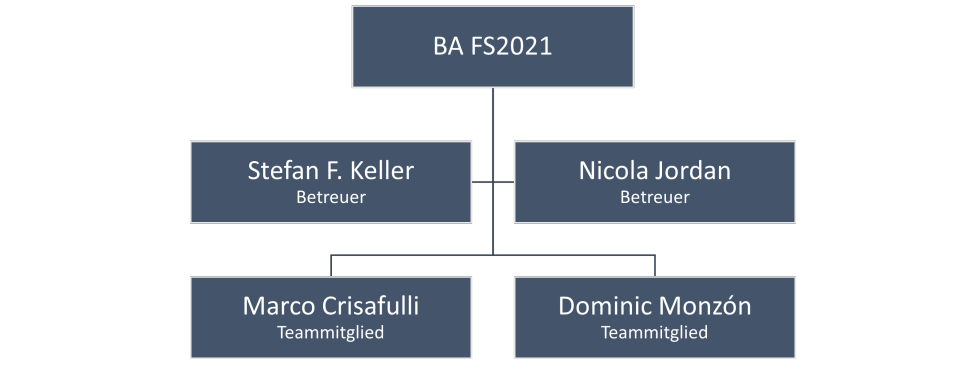


FIGURE D.1: Organizational Structure

Effort Estimation, Timeline, Project Plan

Effort Estimation

The recording of work packages, the effort estimation as well as the time tracking are done in Gitlab and can be seen in full detail under the following link <https://gitlab.com/mcrisafu/mapcomplete/-/issues>. A summary and the charts can be found in appendix E.

Meetings

In addition to short-term, team-internal meetings, the following meetings are held on a regular basis:

- Thursdays 1.15 p.m.: weekly progress meeting focusing on progress since last meeting, obstacles encountered, work planned until next meeting, and decisions to be made.

Project Plan

The project is planned and executed in the period from 22.02. - 18.06.2021. A total of 720 hours are estimated for the execution. This corresponds to an approximate effort of 24 hours per team member per week. Due to the experimental nature of this project, it will be implemented in an agile manner. The project lifecycle consists of the phases Planning & Project Setup, Data Collection & Data Labeling, Training & Debugging and finally Deploying & Testing. These individual phases build on each other, but it is possible to iterate back to earlier phases to make optimizations. Figure D.2 shows the dependencies of the individual phases.

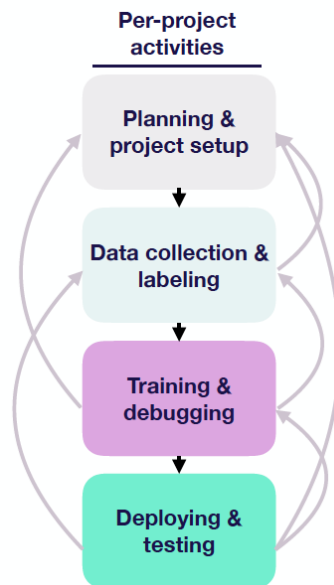


FIGURE D.2: Project Lifecycle
<https://fullstackdeeplearning.com>

In the first phase, the problem is analyzed and the requirements and goals are defined, as well as decisions are made on how resources are to be used. In the second phase, data is collected from various sources, annotated, and prepared for training. The first goal in phase three is to implement a base model early, find and reproduce methods for the problem domain, and improve model performance. The final phase is to test the model in a constrained environment and eventually move it to production. (Tobin et al., 2019)

It is not only optional to iterate back to a previous phase. The missing expertise will make it mandatory to iterate forth and back often, to test different approaches. As deployment is an optional feature most phase changes will be between phase two and three.

Risk Analysis

Table D.3 shows the risks identified during planning. Due to the high uncertainty and the experimental approach of the project, a generous reserve is planned from the beginning and the scope of the work is defined flexibly.

TABLE D.3: Risk Analysis

Nr.	Risk	Damage	Prob.	Consequences
R1	Reduced manpower	high	low	Reduction of scope, delay
R2	Missing know-how	medium	medium	Delay
R3	Wrong estimations	medium	high	Delay, blocked follow up tasks
R4	Insufficient data	high	low	Delay, reduced added value
R5	Inaccurate model	high	medium	Alternative approach
R6	Lack of computing power	medium	low	Additional costs for hardware

Preventative Measures

To mitigate the occurrence of risks described in table D.3, the following preventive measures are taken to mitigate risks with a medium or high probability of occurrence:

- R2** Planning of reserves, regular meetings on the status of work.
- R3** Planning for reserves, effort estimates are made conservatively and revised if necessary.
- R5** Adjust project planning so that multiple iterations are planned to continuously improve the outcome.

Incurred Risks

In the course of the project, the risks listed in table D.4 have occurred.

TABLE D.4: Risks Incurred

Nr.	Damage	Behaviour / Measures
R6	10h	Precalculated data stored

Since only one risk incurred it is explained in more detail. Not the lack of computing power incurred. It was more an underestimation on how much the data would hurt computing power. For one, the PBF of Switzerland was too big to process. It did not load at all. This was no further problem because making a PBF per needed region was simple and anyway the preferred approach. The other more hurtful problem was that for the training area the whole data preparation pipeline took more time than thought. It was fast enough to not invest more resources into the optimization, but it was slow enough to hurt in the sum. And during development it needed to run multiple times. In retrospect, it would have been smart to save the intermediate results earlier or to reduce the area.

Appendix E

Time Tracking

This chapter summarizes and visualizes time tracking. All values are at time of writing and subject to change as the project is not yet finished.

The total work estimated is 537.5 hours. The work done 678.75. The difference of 141.25 hours is primarily because the later issues like documentation and demo were underestimated. Also, the learning task was estimated quite optimistic as seen in the table below.

TABLE E. 1: Five Most Over- and Underestimated Work Items

ID	Title	Estimate	Spent	Diff
58	Write documentation 'technical report'	50.0	104.25	54.25
7	learn about ML	40.00	62.50	22.50
68	Create simple demo	16.00	37.00	21.00
50	TableLearner experiment with different metrics	8.00	23.00	15.00
60	Inspect possibilities for Additional Data	8.00	18.50	10.50
22	Learn hvPlot / basic statistical exploration	16.00	12.00	-4.00
76	Administrative Tasks	48.00	41.50	-6.50
20	Create CI/CD pipeline	8.00	1.00	-7.00
77	Finalize Submission	16.00	8.00	-8.00
14	Base-Line / Base-Model	16.00	7.50	8.50

The following pie chart shows the work done by label. This is quite interesting as one would assume that meetings and an optional feature like demo would use less time in comparison. On the other hand, the meetings were an important and valuable part of the project. Not every hour was already recorded at the time of writing but roughly 25% actual documentation is not that much for a thesis. Maybe it has to do with the fact that a lot of documentation work was done during the experimentation and therefore not labelled correctly.

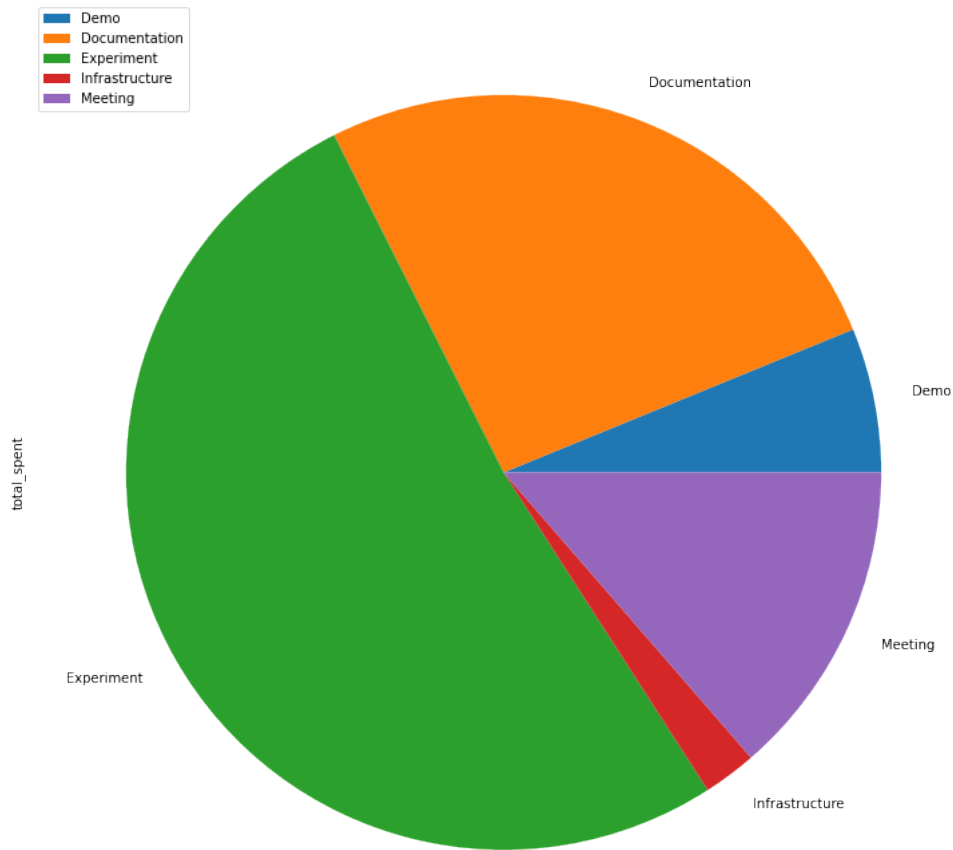


FIGURE E.1: Work by Label

The next two plots show the work done over time. The first cumulated the second separate. This is also interesting. At the beginning of May both authors declined with their hours per day. Not for long but a week “pause” is visible and this without agreement.

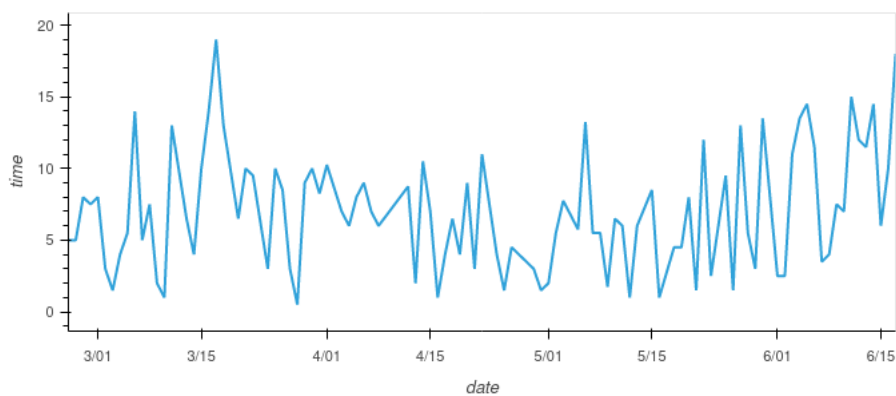


FIGURE E.2: Cumulated Work Over Time

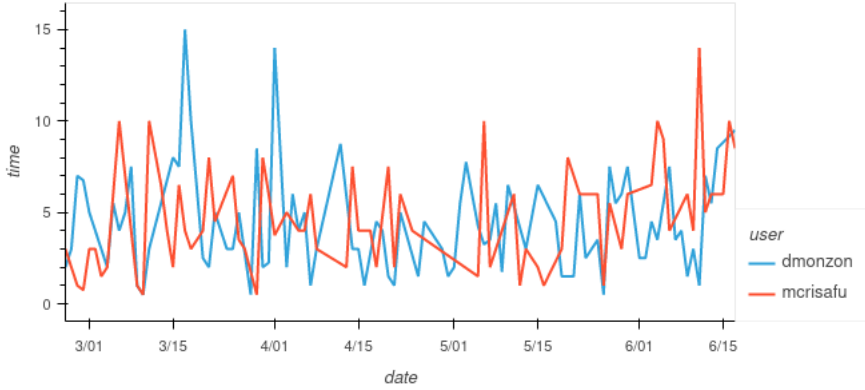


FIGURE E.3: Seperate Work Over Time

The co-authors worked practically the same amount of time. The difference is less than 3%.